МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«БРАТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра математики и физики

УТВЕРЖДАЮ:
Проректор по учебной работе
_____ Е.И. Луковникова
«_____» декабря 2018 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

ЯЗЫКИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

Б1.Б.14

НАПРАВЛЕНИЕ ПОДГОТОВКИ

01.03.02 Прикладная математика и информатика

ПРОФИЛЬ ПОДГОТОВКИ
Инженерия программного обеспечения

Программа академического бакалавриата

Квалификация (степень) выпускника: бакалавр

СОДЕРЖАНИЕ ПРОГРАММЫ **Стр.**

1. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Вид деятельности выпускника

Дисциплина охватывает круг вопросов, относящихся к организационно-управленческому виду профессиональной деятельности выпускника в соответствии с компетенциями и видами деятельности, указанными в учебном плане.

Цель дисциплины

Целью изучения дисциплины является: ознакомление обучающихся с основами теории программирования, развитие навыков работы в различных системах программирования, освоение различных методов, приемов и способов решения задач из различных предметных областей.

Задачи дисциплины

- раскрыть значение дисциплины в общем и профессиональном образовании человека;
- приобретение навыков работы со стандартными и нестандартными алгоритмами решения задач на компьютере;
- сформировать умения и навыки самостоятельного проектирования программ и решения различного рода задач путем применения средств программирования совместно с другими видами программного обеспечения.

Код компетенции	Содержание компетенций	Перечень планируемых результатов обучения по дисциплине
1	2	3
ОПК-3	Способность к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, созданию информационных ресурсов глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям	**знать:** - основные подходы к разработке программ и базовые алгоритмы обработки данных; **уметь:** - разрабатывать программные решения в области прикладного программного обеспечения; **владеть:** – приемами построения алгоритмических и программных решений
ОПК-4	Способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	**знать:** - основные информационно-коммуникационные технологи и требования информационной безопасности **уметь:** – применять информационно-коммуникационные технологии в профессиональной деятельности **владеть:** - культурой применения информационно-коммуникационных технологий

ПК-9	Способность составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы	**знать:** - основы рационального планирования времени и ресурсов, необходимых для работы **уметь:** – разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения **владеть:** - навыками планирования, контроля и оценки результатов собственной деятельности

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина Б1.Б.14 Языки и методы программирования относится к базовой части.

Дисциплина Языки и методы программирования базируется на знаниях, полученных при изучении основных общеобразовательных программ.

Основываясь на изучении дисциплины, Языки и методы программирования представляет основу для изучения дисциплин Базы данных, Системное программирование, Параллельное программирование, Функциональное программирование, Логическое программирование.

Такое системное междисциплинарное изучение направлено на достижение требуемого ФГОС уровня подготовки по квалификации бакалавр.

.

3. РАСПРЕДЕЛЕНИЕ ОБЪЕМА ДИСЦИПЛИНЫ

3.1. Распределение объема дисциплины по формам обучения

Форма обучения	***Курс***	***Семестр***	***Трудоемкость дисциплины в часах***						***Курсовая работа***	***Вид промежуточной аттестации***
			Всего часов (с экз.)	***Аудиторных часов***	***Лекции***	***Лабораторные работы***	***Семинары Практические занятия***	***Самостоятельная работа***		
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Очная	1,2	2,3,4	432	212	53	159	-	184	3КР	Зачет, экзамен
Заочная	-	-	-	-	-	-	-	-	-	-
Заочная (ускоренное обучение)	-	-	-	-	-	-	-	-	-	-
Очно-заочная	-	-	-	-	-	-	-	-	-	-

3.2. Распределение объема дисциплины по видам учебных занятий и трудоемкости

Вид учебных занятий	*Трудо-емкость (час.)*	*в т.ч. в интерактивной, актив-ной, иннова-ционной формах, (час.)*	*Распределение по семестрам, час*		
			2	*3*	*4*
1	**2**	**3**	**4**	**5**	**6**
I. Контактная работа обучающихся с преподавателем (всего)	212	122	72	68	72
Лекции (Лк)	53	22	18	17	18
Лабораторные работы (ЛР)	159	100	54	51	54
Курсовая работа*	+	-	-	+	-
Групповые (индивидуальные) консультации*	+	-	+	+	+
II.Самостоятельная работа обучающихся (СР)	184	-	36	121	27
Подготовка к лабораторным работам	60	-	18	20	22
Подготовка к экзамену в течение семестра	71	-	-	71	-
Подготовка к зачету	23		18	-	5
Выполнение курсовой работы	30	-	-	30	-
III. Промежуточная аттестация экзамен	36	-	-	-	36
зачет	+	-	+	+	-
Общая трудоемкость дисциплины час.	432	-	108	189	135
зач. ед.	12	-	3,0	5,25	3,75

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Распределение разделов дисциплины по видам учебных занятий

- для очной формы обучения:

№ раз-дела и темы	*Наименование раздела и тема дисциплины*	*Трудоем-кость, (час.)*	*Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)*		
			учебные занятия		*самостоя-тельная работа обучаю-щихся*
			лекции	*лабораторные работы*	
1	2	3	4	5	6
1.	**Командно-ориентированное программирование**	**108**	**18**	**54**	**36**
1.1.	Обзор языков програмиирования	7	1	-	6
1.2.	Синтаксис и семантика языка	19	3	10	6
1.3.	Основные алгоритмические конструкции	22	4	12	6
1.4.	Массивы и указатели	22	4	12	6
1.5.	Функции	20	4	10	6
1.6.	Структуры	18	2	10	6
2.	**Объектно-ориентированное программирование**	**189**	**17**	**51**	**121**
2.1	Концепция объектно-ориентированного программирования	23	2	-	21
2.2	Основные конструкции языка C#	33	3	10	20
2.3.	Классы	34	4	10	20
2.4.	Полиморфизм	32	2	10	20
2.5.	Событийно-ориентированное программирование	34	4	10	20
2.6.	Наследование	33	2	11	20
3.	**Динамические струкутуры данных и алгоритмы их обработки**	**99**	**18**	**54**	**27**
3.1	Типы значений и ссылочнные типы	16	2	10	4
3.2.	Универсальные типы	20	4	12	4
3.3.	Линейные структуры	18	2	12	4
3.4.	Деревья	18	4	10	4
3.5.	Табличные структуры	19	4	10	5
3.6.	Коллекции	8	2	-	6
	ИТОГО	**396**	**53**	**159**	**184**

4.2. Содержание дисциплины, структурированное по разделам и темам

№ раздела и темы	*Наименование раздела и темы дисциплины*	*Содержание лекционных занятий*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1	**2**	**3**	**4**
1.	**Командно-ориентированное программирование**		
1.1.	Обзор языков программирования	Классификация языков программирования по степени близости к машинному коду. Классификация языков программирования по стилю программирования. Классификация языков программирования по способу трансляции.	Лекция-беседа (2 часа)
1.2.	Синтаксис и семантика языка	Структура языка программирования. Формальные языки. Грамматики.	Проблемная лекция (2 часа)
1.3.	Основные алгоритмические конструкции	Алгоритмизация. Способы записи алгоритма. Конструкции ветвления. Циклы.	Лекция-беседа (2 часа)
1.4.	Массивы и указатели	Массивы. Определение. Объявление и инициализация массива. Обработка элементов массива. Указатели. Объявление и инициализация указателей. Операции над указателями.	Лекция-беседа (2 часа)
1.5.	Функции	Объявление и описание функций. Прототип функции. Передача аргументов в функцию. Фактические и формальные параметры. Передача аргументов по ссылке. Передача в массивов в функцию.	Проблемная лекция (2 часа)
1.6.	Структуры	Объявление и инициализация структуры. Создание экземпляра структуры. Массивы структур. Доступ к полям структуры. Передача структур в функцию	Проблемная лекция (2 часа)
2.	**Объектно-ориентированное программирование**		
2.1	Концепция объектно-ориентированного программирования	Объектный подход. Полиморфизм. Абстрагирование и инкапсуляция. Наследование и переопределение.	Лекция-беседа (2 часа)
2.2	Основные конструкции языка C#	Типы данных в C#. Динамические переменные. Создание и уничтожение динамических переменных. Циклы и ветвления в C#. Массивы в C#.	Проблемная лекция (4 часа)
2.3	Классы	Классы и объекты. Создание и	-

		уничтожение объекта класса. Конструкторы и деструкторы. Поля класса. Спецификаторы доступа. Статические члены и статические данные.	
2.4	Полиморфизм	Бинарные операторы. Унарные операторы. Перегрузка операции индексации. Перегрузка преобразования типа.	-
2.5	Перехват и обработка событий	Приемники, источники и обработчики события. Функции обратного вызова. Делегаты : сигнатура, объявление и создание объекта. Отправка и перехват событий.	-
2.6	Наследование	Основные принципы построения иерархии классов в C#. Базовый и производный классы. Конструктор производного класса. Создание объекта производного класса. Переопределение методов в производном классе. Влияние спецификаторов доступа на наследуемые члены.	-
2.7	Обработка исключений	Типы исключений. Перехват исключений. Базовый класс System.Exception. Обработка нескольких исключений	-
3.	**Динамические струкутуры данных и алгоритмы их обработки**		
3.1	Типы значений и ссылочнные типы	Ссылки. Ссылочные ипы. Типы значений. Использование ref- и out-параметров. Значение null и Nullable-типы	Проблемная лекция (2 часа)
3.2.	Универсальные типы	Универсальность. Универсальные классы и методы. Синтаксис универсального класса. Ограничения универсальности. Ограничение типа с помощью условия where.	Проблемная лекция (2 часа)
3.3.	Линейные структуры	Стек. Организация стека. Списки. Однонаправленный список. Двунаправленный список. Список с ключом. Очереди.	Проблемная лекция (2 часа)
3.4.	Деревья	Деревья: основные определения. Реализация бинарного дерева. Обход дерева.	
3.5.	Табличные структуры	Словари. Открытое и закрытое хеширование. Алгоритмы хеширования.	
3.6.	Коллекции	Встроенные типы коллекций. Необобщенные и обобщенные коллекции.	

4.3. Лабораторные работы

№ п/п	*Номер раздела дисциплины*	*Наименование лабораторной работы*	*Объем (час.)*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1	**1.**	Конструкции ветвления	8	-
2		Циклы	8	Работа в малых группах (8 час.)
3		Одномерные массивы	8	Работа в малых группах (8 час.)
4		Обработка матриц	10	Работа в малых группах (10 час.)
5		Создание пользовательских функций	10	Работа в малых группах (10 час.)
6		Обработка строк	10	Работа в малых группах (10 час.)
7	**2.**	Основные элементы управления	8	Работа в малых группах (8 час.)
8		Работа с формами	8	Работа в малых группах (8 час.)
9		Обработка списков	8	-
10		Математические функции	9	-
11		Панель инструментов	9	-
12		Таблицы	9	-
13	**3.**	Файлы	10	Работа в малых группах (10 час.)
14		Отображение HTML-файлов	8	-
15		Бестиповые указатели	8	-
16		Связные списки	8	-
17		Графика	10	Работа в малых группах (10 час.)
18		Диаграммы	10	Работа в малых группах (10 час.)
		ИТОГО	**159**	**100**

4.4. Семинары/ практические занятия

учебным планом не предусмотрено

4.5. Контрольные мероприятия: курсовая работа

Курсовая работа выполняется как индивидуальное домашнее задание. Зачтенные работы оформляются и включаются в портфолио обучающегося.

Цель: Углубление знаний в области объектно-ориентированного программирования, формирование навыков самостоятельной работы по дисциплине.

Структура:

1. Титульный лист.
2. Содержание.
3. Введение.
4. Глава 1 (теоретическая).
5. Глава 2 (описание разработки).
6. Заключение.
7. Список использованных источников.
8. Приложения (при необходимости)

Основная тематика: Разработка прикладных программ на объектно-ориентированном языке C#.

Рекомендуемый объем: 25-35 стр.

Выдача задания и защита курсовых работ проводится в соответствии с календарным учебным графиком.

Оценка	Критерии оценки курсовой работы
отлично	Задача на разработку решена полностью. Программа работает устойчиво, в ней предусмотрена обработка ошибок и исключений и - работа содержит грамотно изложенную теоретическую базу с последовательным и аргументированным изложением, обоснованными выводами и предложениями по использованию полученных результатов и - работа оформлена в соответствии с нормативными требованиями, предъявленными к подобным материалам и -работа не содержит грамматических ошибок, опечаток, неаккуратных исправлений и - уникальность текстовой части работы не ниже 60% и - при защите обучающийся четко, ясно, последовательно излагает суть работы, уверенно отвечает на вопросы.
хорошо	Задача на разработку решена полностью, однако программа не обрабатывает возникающие исключения и содержит грамотно изложенную теоретическую базу с последовательным и аргументированным изложением, обоснованными выводами и предложениями по использованию полученных результатов; и/или -при оформлении работы допущены опечатки, некоторые элементы оформления не соответствуют предъявленным требованиям; -уникальность текстовой части работы не ниже 60%;

	- при защите обучающийся показывает знание темы, последовательно излагает суть работы, без особых затруднений отвечает на вопросы.
удовлетворительно	Задача на разработку решена не полностью. Реализованы не менее половины требуемых функций. и/или - работа содержит теоретическую базу, но отличается поверхностным анализом проблем или просто их перечислением без соответствующего анализа, в ней просматриваются непоследовательность изложения и отсутствие описания или анализа собственных результатов, в работе содержатся невнятные выводы и предложения; и/или - при оформлении работы допущены опечатки, некоторые элементы оформления не соответствуют предъявленным требованиям; - уникальность текстовой части работы от 40% до 60% - при защите обучающийся показывает поверхностные знания, на вопросы отвечает неуверенно.
неудовлетворительно	Задание на разработку решено менее, чем на 50% или уникальность текстовой части работы менее 40% или при защите обучающийся обнаруживает незнание большей части темы или совсем не ориентируется в ней, отвечает на вопросы бессистемно, неуверенно, неправильно или же обучающийся вовсе не явился на защиту без уважительной причины.

5. МАТРИЦА СООТНЕСЕНИЯ РАЗДЕЛОВ УЧЕБНОЙ ДИСЦИПЛИНЫ К ФОРМИРУЕМЫМ В НИХ КОМПЕТЕНЦИЯМ И ОЦЕНКЕ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Компетенции / *№, наименование разделов дисциплины*	*Кол-во часов*	*Компетенции*			*Σ комп.*	$t_{ср}$, *час*	*Вид учебных занятий*	*Оценка результатов*
		ОПК		*ПК*				
		3	*4*	*9*				
1	2	3	4	5	6	7	8	9
1. Командно-ориентированное программирование	108	+	+	-	2	54	Лк, ЛР	зачет
2. Объектно-ориентированное программирование	189	-	+	+	2	94,5	Лк, ЛР	КР, зачет
3. Динамические структуры данных и алгоритмы их обработки	99	-	-	+	1	99	Лк, ЛР	экзамен
всего часов	**396**	**54**	**148,5**	**193,5**	**3**	**132**		

6. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Дьяконица С.А. , Семенов Д. С. Основы программирования на языке Си/Си++: Лабораторный практикум. – Братск: Изд-во БрГУ, 2015. – 153 с/
2. Шичкина Ю.А. Создание приложений на языке Visual C# в среде программирования Visual Studio : учебное пособие / Ю. А. Шичкина. - Братск : БрГУ, 2011. - 210 с.

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

№	*Наименование издания*	*Вид заня-тия*	*Количество экземпляров в библиотеке*, шт.	*Обеспечен-ность, (экз./чел.)*
1	2	3	4	5
	Основная литература			
1.	Информатика. Базовый курс : учебник для бакалавров и специалистов / Под ред. С. В. Симоновича. - 3-е изд. - Санкт-Петербург : Питер, 2015. - 640 с. - (Учебник для вузов. Стандарт третьего поколения).	Лк, КР	123 вкючая аналоги	1,0
2.	Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург : Питер, 2014. - 688 с. - (Учебник для вузов. Стандарт третьего поколения)	ЛР	10 включая аналоги	0,5
	Дополнительная литература			
3.	Павловская, Т. А. С/С++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.	КР, ЛР	24	1,0
4.	Программирование и основы алгоритмизации : учебное пособие / В.К. Зольников, П.Р. Машевич, В.И. Анциферова, Н.Н. Литвинов ; Министерство образования и науки Российской Федерации, Федеральное агентство по образованию, Государственное образовательное учреждение высшего профессионального образования «Воронежская государственная лесотехническая академия». - Воронеж : Воронежская государственная лесотехническая академия, 2011. - 341 с. : ил. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=142309	Лк, СР	ЭР	1,0

8. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ» НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1. Электронный каталог библиотеки БрГУ http://irbis.brstu.ru/CGI/irbis64r_15/cgiirbis_64.exe?LNG=&C21COM=F&I21DBN=BOOK&P21DBN=BOOK&S21CNR=&Z21ID=
2. Электронная библиотека БрГУ http://ecat.brstu.ru/catalog
3. Электронно-библиотечная система «Университетская библиотека online» http://biblioclub.ru
4. Электронно-библиотечная система «Издательство «Лань» http://e.lanbook.com
5. Информационная система "Единое окно доступа к образовательным ресурсам" http://window.edu.ru

6. Научная электронная библиотека eLIBRARY.RU http://elibrary.ru

Специальные тематические сайты:

1. Электронный журнал ”Типичный программист” https://tproger.ru .
2. Сайт по программированию https://profssorweb.ru
3. Сайт по программированию https://metanit.com

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Обучающийся должен разработать собственный режим равномерного освоения дисциплины. Подготовка студента к предстоящей лекции включает в себя ряд важных познавательно-практических этапов:

– чтение записей, сделанных в процессе слушания и конспектирования предыдущей лекции, вынесение на поля всего, что требуется при дальнейшей работе с конспектом и учебником;

– техническое оформление записей (подчеркивание, выделение главного, выводов, доказательств);

– выполнение практических заданий преподавателя;

– знакомство с материалом предстоящей лекции по учебнику и дополнительной литературе.

Успешность выполнения лабораторных работ определяется подготовкой к ним. Подготовка к лабораторным работам содержит

- изучение теоретического материала, содержащегося в учебной литературе, изучение лекционного материала,

- знакомство с заданиями на лабораторную работу;

- составление плана выполнения лабораторной работы.

Наиболее продуктивной является самостоятельная работа в библиотеке, где доступны основные и дополнительные печатные и электронные источники.

При выполнении приведенных выше рекомендаций подготовка к зачету сведется к повторению изученного и совершенствованию навыков применения теоретических положений и различных методов решения к стандартным и нестандартным заданиям.

9.1. Методические указания для обучающихся по выполнению лабораторных работ

Лабораторная работа № 1

Конструкции ветвления

Цель работы:

Приобрести навыки реализации разветвляющихся алгоритмов.

Теоретические сведения

Элемент TableLayoutPanel также переопределяет панель и располагает дочерние элементы управления в виде таблицы, где для каждого элемента имеется своя ячейка. Если нам хочется поместить в ячейку более одного элемента, то в эту ячейку добавляется другой компонент TableLayoutPanel, в который затем вкладываются другие элементы.

Чтобы установить нужное число строки столбцов таблицы, мы можем использовать свойства Rows и Columns соответственно. Выбрав один из этих пунктов в окне Properties (Свойства), нам отобразится следующее окно для настройки столбцов и строк:

В поле Size Type мы можем указать размер столбцов / строк. Нам доступны три возможные варианта:

Absolute: задается абсолютный размер для строк или столбцов в пикселях

Percent: задается относительный размер в процентах. Если нам надо создать резиновый дизайн формы, чтобы ее строки и столбцы, а также элементы управления в ячейках таблицы

автоматически масштабировались при изменении размеров формы, то нам нужно использовать именно эту опцию
AutoSize: высота строк и ширина столбцов задается автоматически в зависимости от размера самой большой в строке или столбце ячейки
Также мы можем комбинировать эти значения, например, один столбец может быть фиксированным с абсолютной шириной, а остальные столбцы могут иметь ширину в процентах.
В этом диалоговом окне мы также можем добавить или удалить строки и столбцы. В тоже время графический дизайнер в Visual Studio не всегда сразу отображает изменения в таблице - добавление или удаление строк и столбцов, изменение их размеров, поэтому, если изменений на форме никаких не происходит, надо ее закрыть и потом открыть заново в графическом дизайнере.
Итак, например, у вас имеется три столбца и три строки размер у которых одинаков - 33.33%. В каждую ячейку таблицы добавлена кнопка, у которой установлено свойство Dock=Fill.
В коде динамически мы можем изменять значения столбцов и строк. Причем все столбцы представлены типом ColumnStyle, а строки - типом RowStyle:

```
1    tableLayoutPanel1.RowStyles[0].SizeType = SizeType.Percent;
2    tableLayoutPanel1.RowStyles[0].Height = 40;
3
4    tableLayoutPanel1.ColumnStyles[0].SizeType = SizeType.Absolute;
5    tableLayoutPanel1.ColumnStyles[0].Width = 50;
```

Для установки размера в ColumnStyle и RowStyle определено свойство SizeType, которое принимает одно из значений одноименного перечисления SizeType
Добавление элемента в контейнер TableLayoutPanel имеет свои особенности. Мы можем добавить его как в следующую свободную ячейку или можем явным образом указать ячейку таблицы:

```
1    Button saveButton = new Button();
2    // добавляем кнопку в следующую свободную ячейку
3    tableLayoutPanel1.Controls.Add(saveButton);
4    // добавляем кнопку в ячейку (2,2)
5    tableLayoutPanel1.Controls.Add(saveButton, 2, 2);
```

В данном случае добавляем кнопку в ячейку, образуемую на пересечении третьего столбца и третьей строки. Правда, если у нас нет столько строк и столбцов, то система автоматически выберет нужную ячейку для добавления.

Задание:
Создать приложение со следующими элементами и функциями:
1. Главная форма (рис. 1).
2. При нажатии любой из кнопок автоматически или вручную форма принимает вид рис. 2.
Рис. 1. Главная форма Задание1 Задание2 Введите число строк: число столбцов: Создать: Автоматически 3 3 Вручную Задание1 Задание2 Очистить Дано: Результат: Вычислить Отмена
3. Кнопка «Автоматически» – автоматическое создание мат- рицы заданного размера и автоматическое вычисление. На форме присутствуют надписи, таблицы и кнопка «Отмена».
4. Кнопка «Вручную» – создание матрицы вручную. На форме присутствуют надписи, одна таблица для ввода данных, кнопки «Вычислить», «Очистить», «Отмена». При нажатии кнопки Вычис- лить выводится результат в виде числа для первого задания или в виде таблицы для второго задания.
Задания: 1) найти количество строк, не содержащих ни одного нулевого элемента;
2) поменять местами два произвольных столбца матрицы.

Порядок выполнения:

1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.

3. Протестировать приложение.

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.) .

Дополнительная литература

Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки
1. Опишите синтаксис условного оператора if
2. Как выглядит краткая форма оператора if?.
3. Сформулируйте основные принципы работы с двоичным потоком.
4. Перечислите классы для работы с таблицами

Лабораторная работа № 2

Циклы

Цель работы:
Приобрести навыки реализации алгоритмов, содержащих циклы

Теоретические сведения

Цикл for

Цикл for является циклом с предпроверкой условия выхода. Конструкция for-цикла имеет синтаксис:

for (инициализация; условие; модификаторы) { тело цикла };

Если инициализация включает несколько операторов, то они должны быть разделены запятыми. Выражение может быть опущено в конструкции цикла, но точка с запятой после него должна быть оставлена. В качестве выражения инициализации применяются присваивания или вызовы функций.

Условие формулирует условие продолжения цикла Если условие пропущено, то считается, что его значение всегда истинно. В последнем случае операторы тела цикла будут повторяться до тех пор, пока среди них не встретится break , return или exit().

Модификаторы обычно содержат приращения переменных цикл в виде i++, --j, k+=3, m*=2 … . Если они включает несколько операторов, то они должны быть разделены запятыми.

Цикл while

Цикл while является циклом с предпроверкой условия выхода. Конструкция while-цикла имеет синтаксис:

while (выражение) { тело цикла };

Если при вычислении выражения получается отличный от нуля результат, то выполняется оператор тела цикла, после чего выражение вновь вычисляется. Выражение должно иметь арифметический либо приводимый к нему тип.

Цикл do-while
do
{ тело цикла};
while (условие)
Прерывание вычислений:
continue ; - переход к следующей итерации.

Оператор continue возвращает управление к началу цикла. Следовательно, операторы, стоящие в цикле после continue, не выполняются.

break; - выход из цикла .

В противоположность оператору continue при выполнении break происходит не переход к началу цикла, а выход из тела цикла (т.е. прекращается выполнение не только данного прохода цикла, но и цикла в целом).

return N; - выход из функции

Оператор возврата return реализует механизм выхода из функции. Выполнение функции при этом прерывается, и программное управление передается в точку, из которой была вызвана функция.

exit (N); -выход из программы

С помощью функции exit() можно прервать выполнение программы в любом месте. Такой выход из программы используется при возникновении серьезной ошибки, которая делает дальнейшее выполнение программы бессмысленным или невозможным.

Задание:

1. Составить программу вычисления значений функции $F(x)=2x^2+5$ на отрезке [a; b] с шагом h, где a, b, h вводит пользователь. Результат представить в виде таблицы, первый столбец которой — значения аргумента, второй — соответствующие значения функции.

2. Начав тренировки, спортсмен в первый день пробежал 10 км. Каждый день он увеличивал дневную норму на 10% нормы предыдущего дня. Какой суммарный путь пробежит спортсмен за 7 дней?

3. Вывести на печать 20 случайных целых чисел из диапазона [1;100], определить количество четных в этой последовательности.

4. Напишите программу, тестирующую арифметические способности ученика. Программа предлагает пользователю примеры вида a+b, где a и b - случайные двузначные числа и считывает ответы пользователя. Примеры предлагаются до тех пор, пока пользователь не даст 5 правильных ответов.

Порядок выполнения:

1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.) .

Дополнительная литература

Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки

1. Опишите синтаксис For
2. Как выглядит краткая форма оператора if?.
3. Сформулируйте основные принципы работы с двоичным потоком.
4. Перечислите классы для работы с таблицами

Лабораторная работа № 3

Одномерные массивы

Цель работы:

Приобрести навыки создания и обработки массивов.

Теоретические сведения

Массив - упорядоченная последовательность данных одного типа. Можно говорить о массивах целых чисел, массивов символов и.т.д. Примерами массивов служат матрицы и векторы.

В памяти ЭВМ массивы представляют собой непрерывный участок памяти, с пронумерованными ячейками одинакового размера, обозначаемый одним именем.

Элемент массива (значение элемента массива) – значение, хранящееся в определенной ячейке памяти, расположенной в пределах массива, а также адрес этой ячейки памяти.

Каждый элемент массива характеризуется тремя величинами:

адресом элемента - адресом начальной ячейки памяти, в которой расположен этот элемент;

индексом элемента (порядковым номером элемента в массиве);

значением элемента.

Массив характеризуется следующими основными понятиями:

Адрес массива – адрес начального элемента массива.

Имя массива – идентификатор, используемый для обращения к элементам массива.

Размер массива – количество элементов массива

Размер элемента – количество байт, занимаемых одним элементом массива.

Графически расположение массива в памяти компьютера можно представить в виде непрерывной ленты адресов.

Представленный на рисунке массив содержит q элементов с индексами от 0 до q-1. Каждый элемент занимает в памяти компьютера k байт, причем расположение элементов в памяти последовательное.

Адреса i-го элемента массива имеет значение

$n+k \cdot i$

Адрес массива представляет собой адрес начального (нулевого) элемента массива. Для обращения к элементам массива используется порядковый номер (индекс) элемента, начальное значение которого равно 0. Так, если массив содержит n элементов, то индексы элементов массива меняются в пределах от 0 до n-1.

Объявление и инициализация массивов

Для объявления массива в языке Си используется следующий синтаксис:
тип имя[размер];
или
тип имя[размер]={инициализация};
Здесь тип - тип элементов массива
Размер - количество элементов
Имя - указатель на начало массива

Инициализация представляет собой набор начальных значений элементов массива, разделенных запятыми.

```
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9};  // массив a из 10 целых чисел
```

Если массив проинициализирован при объявлении, то константные начальные значения его элементов указываются через запятую в фигурных скобках. В этом случае количество элементов в квадратных скобках может быть опущено.

```
int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
```

При обращении к элементам массива индекс требуемого элемента указывается в квадратных скобках [].

Однако часто требуется задавать значения элементов массива в процессе выполнения программы. При этом используется объявление массива без инициализации. В таком случае указание количества элементов в квадратных скобках обязательно.

```
int a[10];
```

Для задания начальных значений элементов массива очень часто используется параметрический цикл:

```
 #include <stdio.h>
main() {
const int n=10;
  int a[10]; // объявлен массив a  из 10 элементов
  int i;
  // Ввод элементов массива
  for(i=0; i<n; i++)  {
     printf("a[%d] = ", i);
     scanf("%d", &a[i]);
    }
    for(i=0;i<n;i++)  {
      printf("%d ",a[i]); // пробел в формате печати обязателен
    }
}
```

Задание:
Создать одномерный массив из n элементов последовательности $\{x_i\}$, где элемент x_i вычисляется по формуле $x_i = \frac{i(i+1)}{2}$. Найти сумму элементов массива. Число n запросить у пользователя.

2. Создать одномерный массив из n случайных целых чисел из диапазона [10; 20]. Найти и напечатать:
а) Значение максимального элемента;
б) Значение и индекс(ы) минимального элемента;
в) Значение элемента наиболее близкого к среднему арифметическому элементов массива;

г) Количество четных элементов.

3. Создать массив из n (n<10) элементов. И
а) "перевернуть" массив, записав его элементы в него же в обратном порядке;
б) сжать массив, удалив из него все нулевые элементы;
в) создать второй массив и записать в него все элементы первого без повторов.

Порядок выполнения:

1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.

Дополнительная литература

Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки
1. Что представляет собой поток (stream)?
2. Сформулируйте основные принципы работы с массивами.
3. Сформулируйте основные принципы работы с двоичным потоком.
4. Перечислите классы для работы с таблицами

Лабораторная работа № 4

Обработка матриц

Цель работы:
Приобрести навыки создания матриц различных размеров, освоить алгоритмы действий над матрицами.

Теоретические сведения

В языке Си могут быть также объявлены многомерные массивы. Отличие многомерного массива от одномерного состоит в том, что в одномерном массиве положение элемента определяется одним индексом, а в многомерном — несколькими. Примером многомерного массива является матрица.

Общая форма объявления многомерного массива
тип имя[размерность1][размерность2]...[размерностьm];
int a[2][3];

Элементы многомерного массива располагаются в последовательных ячейках оперативной памяти по возрастанию адресов. В памяти компьютера элементы многомерного массива располагаются подряд, например массив, имеющий 2 строки и 3 столбца

Инициализация многомерных массивов

Значения элементов многомерного массива, как и в одномерном случае, могут быть заданы константными значениями при объявлении, заключенными в фигурные скобки {}. Однако в этом

случае указание количества элементов в строках и столбцах должно быть обязательно указано в квадратных скобках [].

Пример

```
#include <stdio.h>
int main() {
  int a[2][3]={1, 2, 3, 4, 5, 6};
  printf("%d %d %d\n", a[0][0], a[0][1], a[0][2]);
  printf("%d %d %d\n", a[1][0], a[1][1], a[1][2]);
  getchar();
  return 0;
 }
```

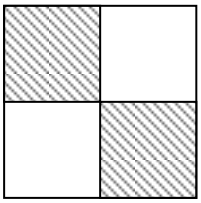
Однако чаще требуется вводить значения элементов многомерного массива в процессе выполнения программы. С этой целью удобно использовать вложенный параметрический цикл.

Пример

```
#include <stdio.h>
int main() {
  int a[2][3]; // массив из 2 строк и 3 столбцов
  int i, j;
  // Ввод элементов массива
  for(i=0; i<2; i++)  // цикл по строкам
   {
    for(j=0; j<3; j++) // цикл по столбцам
     {
       printf("a[%d][%d] = ", i,j);
      scanf("%d", &a[i][j]);
     }
   }
  // Вывод элементов массива
  for(i=0; i<2; i++)  // цикл по строкам
   {
     for(j=0; j<3; j++) // цикл по столбцам
       {
         printf("%d ", a[i][j]);
       }
     printf("\n"); // перевод на новую строку
   }
}
```

Задание

1. Создать массив размера nxn, в котором все элементы, расположенные в заштрихованной области (рис 1.) , равны 0, а остальные — 1.



2. В матрице размера nxm из всех минимальных элементов строк найти максимальный. вывести его значение и координаты.
3. Создать две квадратные матрицы со случайными элементами. Найти их произведение.
4. Написать программу, генерирующую таблицу квадратов двузначных чисел.
по строкам таблицы расположены квадраты элементов с одинаковым значением разряда десятков; по столбцам — единиц.

Порядок выполнения:

1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.

Дополнительная литература

Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки
1. Что представляет собой поток (stream)?
2. Сформулируйте основные принципы работы с массивами.
3. Сформулируйте основные принципы работы с двоичным потоком.
4. Перечислите классы для работы с таблицами

Лабораторная работа № 5

Создание пользовательских функций

Цель работы:
Приобрести навыки создания пользовательских функций, освоить алгоритмы действий над матрицами.

Теоретические сведения

Функция - это именованная часть кода, обладающая своим потоком управления и своей областью памяти.

Функция принимает аргументы из вызывающего её блока и возвращает одно результирующее значение в этот блок.

Функции, которые мы будем создавать сами, обычно требуют объявления прототипа. Прототип дает основную информацию о структуре функции: он сообщает компилятору, какое значение функция возвращает, как функция будет вызываться, а также то, какие аргументы функции могут быть переданы.

Прототип функции:
тип_ВЗ имя (тип1 арг1, тип2 арг2, …, типN аргN); // точка с запятой!
где
тип_ВЗ -тип возвращаемого значения;
имя - имя функции;
типN - тип аргумента N
аргN - имя фиктивного аргумента N
Например:
int mult (int x, int y);

Этот прототип сообщает компилятору, что функция принимает два аргумента, в качестве целых чисел, и что по завершению работы функция вернет целое значение. Обязательно в конце прототипа необходимо добавлять точку с запятой. Без этого символа, компилятор, скорее всего, подумает, что вы пытаетесь написать собственно определения функции.

Чтобы объяснить компилятору. что нужно делать в функции, программист пишет её определение:

тип_ВЗ имя (тип1 арг1, тип2 арг2, …, типN аргN) {
тело функции
}
здесь
тело функции - действия, выполняемые функцией

Когда программист фактически определяет функцию, он начнет с прототипа, но точку с запятой уже ставить не надо. Сразу после прототипа идет блок с фигурными скобочками и с кодом, который функция будет выполнять

Вызов функции имеет вид:
переменная=имя (арг1, арг2, …, аргN);
где

- переменная - переменная, в которую сохраняется возвращенное значение. Эта переменная должна быть того же типа, который указан в определении функции (тип_ВЗ)
- арг1, арг2, …, аргN - фактические параметры функции или их фактические значения.

Если не предполагается возврат значения из функции, то в качестве типа возвращаемого значения указывается void.

Пример 1. Напишем функцию, которая переводит градусную меру угла (g) в радианную(r).

Для этого воспользуемся формулой $$r = \frac{g \cdot \pi}{180}$$
Работа с функцией будет содержать три этапа:
1) Объявление прототипа функции
2) Обращение к функции из main
3) Определение функции

Функция GradToRadian в программе переводит градусную меру угла в радианную.

```
#include <stdio.h>
#include<math.h>
/* ------------------------------------------------------------------------*
*   Объявление прототипа функции:                                         */
float GradToRadian(float alpha);  // точка с запятой
/*------------------------------------------------------------------------*
*   Главная функция программы:                                            */
main(){
   float g,r;  /* g =градусная мера угла, r=радианная мера */
   printf("Введите градусы:  ");
   scanf("%f", &g);
   /* Вызов функции GradToRadian      */
   r = GradToRadian ( g );
   printf("\n Радианная мера этого угла равна %2.2f  рад. \n", r);
   printf("\n Синус этого угла равен %2.2f  рад. \n", sin(r));
}
/* ------------------------------------------------------------------------*
  Определение функции                                                     */
float GradToRadian(float alpha) { //без точки с запятой
  float result=alpha*3.14/180;
  return result; // возвращаемое значение
}
```

```
bash-3.1$ cc func3.c -lm
bash-3.1$ a.out
Введите градусы:  30

 Радианная мера этого угла равна 0.52  рад.

 Синус этого угла равен 0.50  рад.
bash-3.1$
```

Terminal

Функции, обрабатывающие массивы или строки
Для передачи массива в качестве аргумента в функцию, в списке аргументов пишут указатель на массив и его размер. То же самое относится к строкам.
Объявление функции, выводящей на печать массив massive:

```
void print(float massive[], int n);//эти объявления эквивалентны
void printt(float *massive, int n);
```

Объявление функции, которая ищет в строке, состоящей из нескольких слов, номер первого слова, содержащего символ a.

```
int find(char srting[][], chat a, int m, int n); //эти объявления эквивалентны
int find(char *srting[], chat a, int m, int n);
int find(char **string,char a,  int m, int n);
```

В качестве примера рассмотрим функцию, которая преобразует строку, переводя все её символы из нижнего регистра в верхний:

```
#include <string.h>
#include <ctype.h>
 void upper(char *string   ,  int n);

 main() {
   char s[]="abcdefghijklmnoprst";
   upper(s, strlen(s)) ;
   …
 }
/************************************************/
 void upper(char *string   ,  int n) {
    int i;
    for(i=0; string[i]; ++i)  {
       string[i] = toupper(string[i]);
       printf("%c", string [i]);
     }
 }
```

Передача имени функции в качестве параметров другой функции

В функцию можно передать указатель на другую функцию

тип function (тип (*указатель) (типы аргументов),…)

Все передаваемые функции должны иметь одинаковые прототипы (за исключением имени)

```
#include <stdio.h>
#include <stdlib.h>

int qw(int x, int a){
return x*x+a;}
int cube (int x, int a){
return x*x*x-a;}
int rnd(int a, int b){
return rand()%a+b;}

void generation(int massive[], int n, int(*func)(int,int ), int b){
int i;
for (i=0;i<n;i++)
   massive[i]=func(i,b);
}

void print(int massive[], int n){
int i;
for (i=0;i<n;i++)
   printf("%d ",massive[i]);
   printf("\n");
}
main()
```

```
{   int n=5;
    int a[n],b[n],c[n];
    printf("First mass:\n");
    generation (a,n,qw,0);
    print(a,  n);
     generation (b,n,cube,0);
    print(c,  n);
    generation (c,n,rnd,0);
    print(c,  n);}
```

Задание

1. Написать функции для преобразования декартовых координат в полярные и обратно. Продемонстрировать их вызов в программе.
2. Напишите функцию, которая находит площадь прямоугольника. Напишите программу, которая использует эту функцию. Из 10 прямоугольников программа находит тот, у которого площадь наибольшая.
3. Написать функцию Ugol для вычисления угла в треугольнике. Напишите программу, которая определяет все углы в треугольнике, вершины которого вводит пользователь.
4. Напишите функцию, которая вычисляет расстояние между двумя точками на плоскости. Напишите программу, которая использует эту функцию. Программа определяет для N введенных точек две наиболее близкие.
5. Напишите функцию, которая находит сумму цифр целого числа N ($N<2^{15}$). Напишите программу, которая использует эту функцию. Программа обрабатывает массив из K целых чисел и печатает для каждого из них сумму цифр.

Порядок выполнения:

1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:

Отчет по работе содержит:

1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.

Дополнительная литература

Павловская, Т. А. С/С++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки

1. Что представляет собой поток (stream)?
2. Сформулируйте основные принципы работы с массивами.
3. Сформулируйте основные принципы работы с двоичным потоком.
4. Перечислите классы для работы с таблицами

Лабораторная работа № 6

Обработка строк

Цель работы:

Освоить способы работы со строками, изучить функции и алгоритмы их обработки.

Теоретические сведения

Символьная константа - это символ (единственный), заключенный в одиночные кавычки, как, например, 'X'. Значением символьной константы является численное значение этого символа в машинном представлении набора символов. Все символы упорядочены в соответствии с принятым в ПК коде (например ASCII). При этом порядковый номер символов называется кодом (например, код латинского символа 'A ' равен 65; символа '3' равен 51). Для символьных данных не определены никакие арифметические операции, но они могут сравниваться по своим кодам, участвовать в чтении, печати, операторах присваивания.

Строка - это последовательность (массив) символов, заключенная в двойные кавычки, заканчивающаяся нуль-символом (\0 – символ с кодом равным нолю). По положению нуль-символа определяется фактическая длина строки. Например, длина строки char text[ ] = "Моя программа!" равна 14 байт (включая пробел).

Для форматного ввода и вывода символьных констант используется спецификатор %c, строк - %s и специальные функции: getchar(),gets()- ввод и putchar(), puts()-вывод. Библиотека Си содержит функции обработки строк, прототипы которых определяются в заголовочном файле string.h. Например:

strlen(str) – определяет длину строки str;

strcat(str1,str2) - сцепление строк в порядке их перечисления;

strncat(str1,str2,kol) – приписывает kol символов строки str2 к строке str1;

strcmp(str1,str2) – сравнивает две строки str1 и str2 и возвращает 0, если они одинаковы; результат отрицателен, если str1<str2 и положителен, если str1>str2;

strncmp(str1, str2, kol) – сравниваются части строк str1 и str2 из kol символов. Результат равен 0, если они одинаковы.

strcpy(str1,str2) – копирует строку str2 в строку str1.

Про коды

Каждому символу соответствует свой ASCII код. Узнать код просто: достаточно вывести символ в числовом представлении.

Пример:

```
char symb='a';                  // Это символ 'a'
printf("symbol=%c", symb);   // Печатаем его
printf("ASCII =%d", symb);      //Печатаем ASCII код
```

Про шифрование.

Одним из простейших шифров является шифр Цезаря. Его суть состоит в том, что буквы шифруемого текста сдвигаются на некоторое число позиций по алфавиту. Сдвиг происходи циклически, те за буквой 'a' следовала 'z' (римляне использовали латынь, если что). Например, при сдвиге на 2 позиции буква 'a' преобразуется в 'c',

'b' в 'd', а 'z' в 'b'. Величина сдвига называется ключом шифра.

Математически операция сдвига описывается так.

Каждой букве ставим в соответствие число: a – 1, b – 2, … , z – 26,. Далее индекс символа складывается с номером в сообщении по модулю 26, а результат сложения представляется в системе счисления с основанием 26 (0, 1, …, P в верхнем регистре).

Вычислить номер буквы в алфавите можно, используя тот факт, что для ЭВМ буква и её номер -это одно и тоже:

```
int n;
char letter;
…
n=letter-'a'+1;
```

Задание

1. Контрольная сумма

При передаче и хранении данных используют контрольные суммы. В общем виде контрольная сумма представляет собой некоторое значение, вычисленное по определённой схеме на основе кодируемого сообщения. Проверочная информация при систематическом кодировании приписывается к передаваемым данным. На принимающей стороне абонент знает алгоритм вычисления контрольной суммы: соответственно, программа имеет возможность проверить корректность принятых данных. Существуют различные алгоритмы для создания контрольных сумм.

Один из простейших алгоритмов строится на сложении всех байт сообщения по модулю 8. Фактически складываются коды всех символов строки и вычисляется остаток суммы от деления на 8.

Напишите программу, которая вычисляет контрольную сумму введенного пользователем сообщения.

Входные данные:

строка символов - текст сообщения. Также может содержать пробелы, и знаки препинания. Длина строки не должна превышать 100 символов

Выходные данные : целое число из диапазона [0;7]- контрольная сумма.

2. Шифр Цезаря

Напишите программу, которая зашифровывает послание на английском языке шифром Цезаря. Пробелы и знаки препинания оставить как прежде.

Входные данные:

1) строка символов - текст сообщения из прописных латинских букв. Также может содержать пробелы, и знаки препинания. Длина строки не должна превышать 100 символов

2) ключ - целое число из диапазона [-26; 26]

Выходные данные : Зашифрованное сообщение.

3. Шифр Гронсфельда

Напишите программу, которая зашифровывает послание на английском языке шифром Гронсфельда. Ключ содержит до 10 целых чисел . Пробелы и знаки препинания оставить как прежде.

Входные данные:

1) строка символов - текст сообщения из прописных латинских букв. Также может содержать пробелы, и знаки препинания. Длина строки не должна превышать 100 символов

2) Целое число N -длина ключа (N<11)

3) Ключ - набор целых чисел из диапазона [0; 26]

Выходные данные : Зашифрованное сообщение

Порядок выполнения:

1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:

Отчет по работе содержит:

1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с

Дополнительная литература

Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки

1. Что представляет собой поток (stream)?
2. Сформулируйте основные принципы работы с массивами.
3. Сформулируйте основные принципы работы с двоичным потоком.
4. Перечислите классы для работы с таблицами

Лабораторная работа № 7

Основные элементы управления

Цель работы :

Приобрести навыки работы с элементами управления: форма, текстовое поле, радиокнопка, переключатель, панель, надпись.

Теоретические сведения

Классы, представляющие графические элементы управления, находятся в пространстве имен System.Windows.Forms. С их помощью обеспечивается реакция на действия пользователя в приложении Windows Forms. Классы элементов управления связаны между собой достаточно сложными отношениями наследования. Класс Control, как общий предок, обеспечивает все производные классы общим набором важнейших возможностей. В числе этих возможностей можно перечислить события мыши и клавиатуры, физические размеры и местонахождение элемента управления (свойства Height, Width, Left, Right, Location), установку цвета фона и цвета переднего плана, выбор шрифта и т.п.

При создании приложения можно добавить элементы управления на форму при помощи графических средств Visual Studio. Обычно достаточно выбрать нужный элемент управления в окне ToolBox и поместить его на форме. Visual Studio автоматически сгенерирует нужный код для формы. После этого можно изменить название элемента управления на более содержательное (например, вместо button1, предлагаемого по умолчанию, - buttonPrimer) Visual Studio позволяет не только размещать на форме элементы управления, но и настраивать их свойства. Для этого достаточно щелкнуть на элементе управления правой кнопкой мыши и в контекстном меню выбрать Properties (Свойства).

Все изменения, которые необходимо произвести в открывшемся окне (рисунок 1), будут добавлены в код метода InitializeComponents().

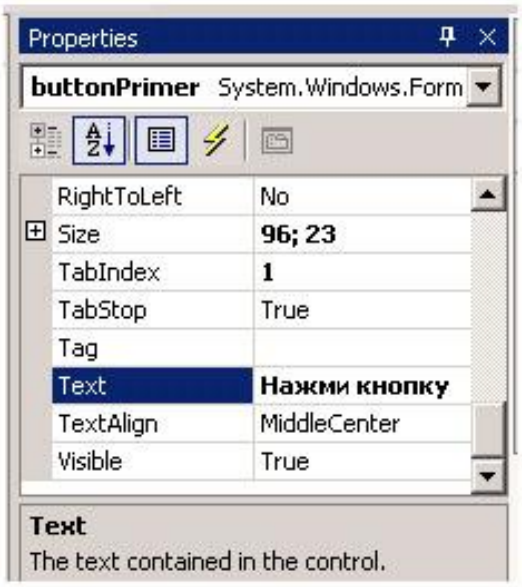


Рис. 1 Настройка элементов управления средствами Visual Studio

То же самое окно позволяет настроить не только свойства данного элемента управления, но и обработку событий этого элемента. Перейти в список событий можно при помощи кнопки  в закладке Properties (рисунок 2). Можно выбрать в списке нужное событие и рядом с ним сделать двойной щелчок или ввести имя метода или выбрать метод из списка.

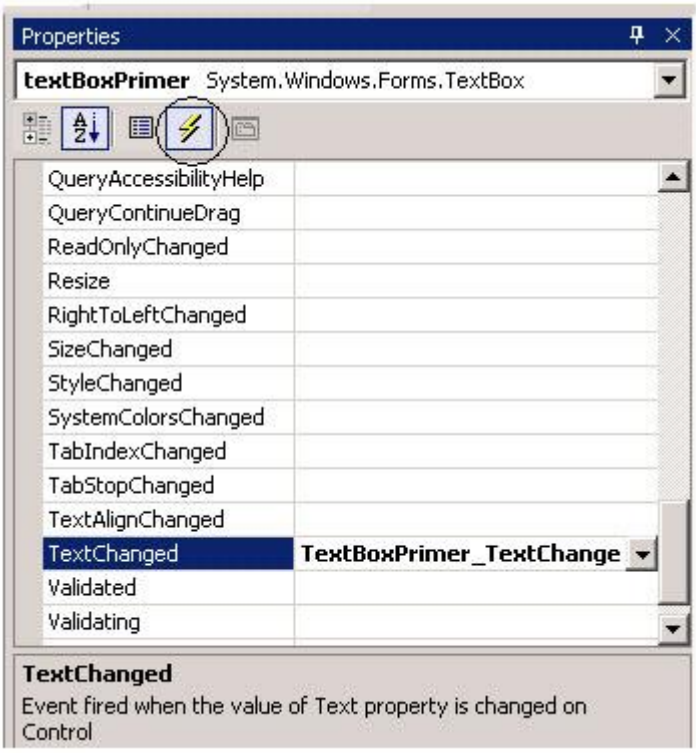


Рис. 2. Настройка обработчиков событий
После задания имени метода или двойного щелчка Visual Studio сгенерирует заготовку для обработчика события.

Запуск приложения
Чтобы запустить приложение в режиме отладки, нажмем на клавишу F5 или на зеленую стрелочку на панели Visual Studio. После этого запустится наша форма с одинокой кнопкой. И если мы нажмем на кнопку на форме, то нам будет отображено сообщение с приветствием. После запуска приложения студия компилирует его в файл с расширением exe. Найти данный файл можно, зайдя в папку проекта и далее в каталог bin/Debug или bin/Release
Задание
Создать проект из одной формы:
При нажатии кнопки Ввести в правой панели должен появляться текст с именем, введенным пользователем в левой панели, и выбранными увлечениями. При повторном нажатии кнопки «Ввести» старый текст в правой панели должен удаляться и выводиться новый.

Порядок выполнения:
1. Изучите теоретический материал по теме лабораторной работы;
2. Создайте проект Windowx Form

3. Выполните действия по созданию и заполнению формы приложения элементами управления
4. В коде программы опишите логику приложения
5. Протестируйте программу
6. Оформите отчет

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.

Дополнительная литература

Павловская, Т. А. С/С++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки
1. Какие типы данных используются в Си?
2. Для чего нужно указывать тип переменной?
3. Что выполняет директива include<stdio.h>?
4. Какой командой компилируется программа в gcc?
5. Опишите общую структуру программы

Лабораторная работа № 8

Работа с формами

Цель работы: Приобрести навыки обработки данных, полученных из формы, и отображения результатов.

Теоретические сведения

Создание формы Windows Forms

Для приложения Windows могут понадобиться несколько форм, помимо основной.

В .NET Framework можно легко добавлять диалоговые окна, экраны запуска и другие формы поддержки. Чтобы добавить форму Windows Forms, которая наследуется из класса Form, выполните необходимые действия.

1. Выберите команду Форма Windows одним из следующих способов: □в окне Обозреватель решений в контекстном меню к имени проекта выберите команду Добавить и затем команду Форма Windows; □в программном меню Файл выберите команду Добавить форму Windows; 44 Добавить форму Windows; □на панели инструментов нажмите кнопку Добавить новый элемент: и далее выберите команду Добавить форму Windows: □ с помощью комбинации клавиш Ctrl+Shift+A. В результате откроется окно Добавление нового элемента
2. В области Шаблоны выберите Форма Windows.
3. В поле Имя введите имя новой формы.
4. Нажмите кнопку Добавить

Исключение формы из проекта Чтобы исключить форму из проекта, в окне Обозреватель решения в контекстном меню к файлу имя_формы.cs выберите команду Исключить из проекта. Примечание. После выполнения команды форма будет исключена из проекта, но все файлы для этой формы сохранятся в папке проекта и поэтому форма может быть успешно восстановлена в будущем в данном проекте или добавлена в другой проект

Удаление формы из проекта Чтобы удалить форму из проекта, в окне Обозреватель решения в контекстном меню к файлу имя_формы.cs выберите ко- манду Удалить. Внимание! Не игнорируйте сообщение об удалении. После нажатия кнопки Ок в выданном сообщении

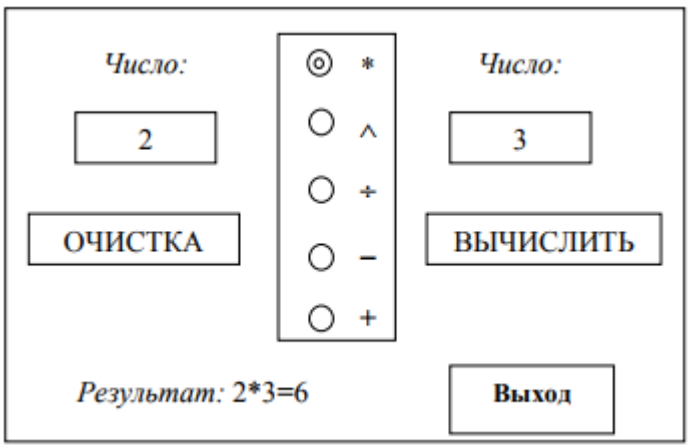
файлы формы будут не только исключены из проекта, они будут физически удалены без возможности восстановления
Добавление формы в проект Чтобы добавить форму в проект, выполните следующие действия:
1. Выберите в меню Проект команду Существующий элемент.
2. В открывшемся окне Добавление существующего элемента выберите файл формы имя_формы.cs.
3. Нажмите кнопку Добавить. Примечание. Файл будет добавлен вместе со вспомогательными файлами настройки и другими метаданными.
Настройка формы
Все действия по изменению внешнего вида формы произво- дятся с помощью команд, отображаемых в окне Свойства в рабочей области проекта. Для отображения окна Свойства необходимо вы- брать в контекстном меню к форме команду Свойства. В результате открывается окно со свойствами формы. В верхней строке окна Свойства жирным шрифтом прописы- вается имя компонента, свойства которого отображаются в этом окне. Справа от имени отображается класс, которому принадлежит данный компонент. Внимание! В конструкторе форм можно изменять размеры формы, но нельзя менять ее местоположение. Чтобы отобразить форму Windows Forms, в конструкторе дважды щелкните форму в Обозревателе решений.

Задание

Создать приложение со следующими формами: 1. Титульная форма:

По кнопке «Далее» должен осуществляться переход на рабо- чую форму.

Титульная форма должна скрываться. Свойства титульной формы: жесткие границы, модальная.



2. Рабочая форма с калькулятором следующего вида: По кнопке Очистка текстовые поля должны очищаться, а пе реключатель устанавливаться в пункт «*».

Порядок выполнения:

1. Изучите теоретический материал по теме лабораторной работы;
2. Создайте проект Windowx Form
3. Выполните действия по созданию и заполнению форм приложения элементами управления
4. В коде программы опишите логику приложения
5. Протестируйте программу
6. Оформите отчет

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.) и предыдущие издания.

Дополнительная литература

Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки
1. Какие виды форм можно создать в проекте?
2. Как объявляется переменная, описывающая класс формы?
3. Что такое модальная форма?
4 Как изменить границы формы?
5. Что произойдет, если закрыть главную форму прложения?

Лабораторная работа № 9

Обработка списков

Цель работы:

Приобрести навыки обработки текстовой информации, приобрести более глубокие знания компонентов.

Краткие теоретические сведения

Типы окон и операции над ними
Интегрированная среда разработки содержит два основных типа окон: окна инструментов и окна документов. В поведении этих окон имеются некоторые различия. Окна инструментов Окна инструментов перечислены в меню Вид. Они соответствуют текущему приложению и его надстройкам. Можно настроить следующие параметры окон инструментов в интегрированной среде разработки:
● автоматическое отображение и скрытие;
● объединение с другими окнами инструментов для создания групп вкладок;
● закрепление по границам интегрированной среды разработки;
● плавающее окно.
Окна документов
Окна документов создаются автоматически при открытии или создании файлов и других элементов. Список открытых окон документов отображается в меню Окно в текущей последовательно сти (верхнее окно указано в списке первым).
Группы вкладок
Группы вкладок упрощают организацию ограниченного рабо чего пространства при наличии двух или более открытых докумен тов в интегрированной среде разработки. Можно организовать несколько окон документов и средств в вертикальные или горизон тальные

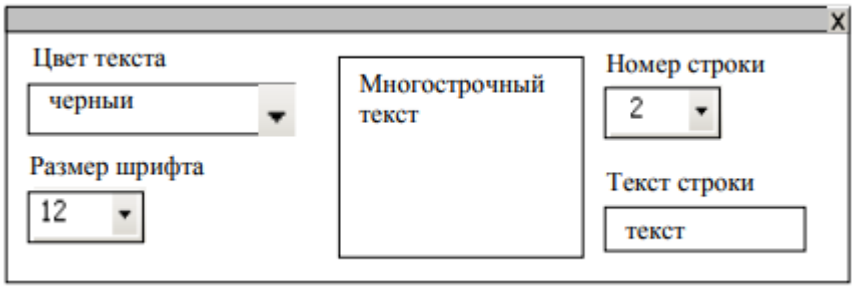
группы вкладок с помощью меню Окно и легко перено сить документы из одной группы в другую.
Разделенные окна
Если требуется просмотреть или изменить два различных ме ста в документе, можно использовать разделение окон. Чтобы раз делить документ на две области с независимой прокруткой, выбе рите команду Разделить в меню Окно. Чтобы восстановить единое отображение, выберите команду Снять разделение в меню Окно.
Чтобы закрепить окно инструментов:
1. Щелкните окно инструментов, которое требуется закрепить.
2. Перетащите окно к середине окна интерфейса IDE. Появится маркер в виде ромба. Четыре стрелки ромба указы вают на четыре стороны панели редактирования.
3. Когда перетаскиваемое окно достигнет нужного расположе ния, наведите указатель на соответствующую часть ромба-маркера. Указанная область будет затемненной.
4. Чтобы закрепить окно в этом положении, отпустите кнопку мыши.
Например, если окно Обозреватель решений закреплено на правой стороне среды разработки, а его необходимо закрепить на левой стороне, то это окно следует перетащить в центр среды разработки и навести указатель на самую левую стрелку ромба и отпустить кнопку мыши.
Чтобы закрепить окно инструментов:
1. Щелкните окно инструментов, которое требуется закрепить.
2. Перетащите окно к середине окна интерфейса IDE. Появится маркер в виде ромба. Четыре стрелки ромба указы вают на четыре стороны панели редактирования.
3. Когда перетаскиваемое окно достигнет нужного расположе ния, наведите указатель на соответствующую часть ромба-маркера. Указанная область будет затемненной.
4. Чтобы закрепить окно в этом положении, отпустите кнопку мыши. Например, если окно Обозреватель решений закреплено на правой стороне среды разработки, а его необходимо закрепить на левой стороне, то это окно следует перетащить в центр среды разработки и навести указатель на самую левую стрелку ромба и отпустить кнопку мыши.
Списки ComboBox
Элемент управления ComboBox (Windows Forms) используется для вывода данных в раскрывающемся поле со списком. По умолчанию элемент управления ComboBox отображается в виде двух частей: верхняя часть представляет собой текстовое поле, в которое пользователь может ввести элемент списка. Вторая часть представляет собой список элементов, один из которых поль зователь может выбрать. Свойство SelectedIndex возвращает целочисленное значение, соответствующее выбранному элементу списка. Выбранный эле мент можно изменить программными средствами, изменив в коде значение SelectedIndex; соответствующий элемент списка появится в текстовом поле поля со списком. Если выбранных элементов нет, значение SelectedIndex равно –1. Если в списке выбран первый эле мент, значение SelectedIndex равно 0. Свойство SelectedItem анало гично свойству SelectedIndex, но возвращает сам элемент, обычно в виде строкового значения. Свойство Count отражает число эле ментов в списке, а значение свойства Count всегда на единицу больше максимально возможного значения свойства SelectedIndex, поскольку для свойства SelectedIndex индексация ведется от нуля. Чтобы добавить или удалить элементы в элементе управления ComboBox, используйте метод Add (добавление элемента в конец списка), Insert (вставка элемента в середину списка), Clear (очистка списка) или Remove (удаление элемента из списка). Кроме того, можно добавить элементы в список с помощью свойства Items во время разработки. Примечание. Все вышеперечисленные свойства и методы до ступны только в коде. В окне Свойства они отсутствуют
Создание списка Чтобы создать список, выполните следующие действия:
1. Добавьте на форму элемент ComboBox.
2. Добавьте в список элементы одним из способов:
● С помощью панели Задачи, которая открывается при щелчке по треугольнику в верхнем правом углу элемента управления ComboBox. В панели Задачи выберите команду Правка элементов.

● С помощью щелчка по символу в свойстве Items элемента управления ComboBox. В результате откроется окно Редактор коллекции строк для правки элементов списка.
3. Введите элементы списка по одному на строку.
4. Нажмите ОК.
5. Введите в качестве значения свойства Text название элемен та, который должен быть виден в начальном состоянии списка в рабочем приложении. Например, пусть в данном примере это элемент: «Синий».
6. Создайте обработчик события выбора элемента списка SelectedIndexChanged в окне Свойства в режиме События. Пример. При выборе цвета из списка должен меняться соот ветственно цвет текста на форме. Щелкните здесь, чтобы открыть панель Задачи

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
 switch (comboBox1.SelectedIndex)
{
 case 0: glob.frm2.ForeColor = Color.Red; break; case 1: glob.frm2.ForeColor = Color.Blue; break;
case 2: glob.frm2.ForeColor = Color.Green; break;
}
;
}
```

Задание

Создайте приложение:



Описание:
1. В разделе «Цвет текста» выбирается цвет текста, печатаемо- го в центральной части окна. Значение по умолчанию – черный. Всего не менее трех цветов.
2. В разделе «Размер шрифта» выбирается размер шрифта для текста, печатаемого в центральной части окна. Значение по умолчанию – 12 пт. Минимальное значение – 8 пт. Максимальное значение – 20 пт. Значения можно выбирать только из списка и нельзя вводить с клавиатуры.
3. В разделе «Номер строки» выбирается номер строки текста, которая должна отобразиться в разделе «Текст строки». Нумерация с единицы. Значение по умолчанию – 1. Обеспечить корректность верхних и нижних границ

Порядок выполнения:

1. Изучите теоретический материал по теме лабораторной работы;
2. Создайте проект Windowx Form
3. Выполните действия по созданию форм приложения
4. В коде программы опишите логику приложения
5. Протестируйте программу
6. Оформите отчет

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;

2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.) и предыдущие издания.

Контрольные вопросы для самопроверки

1. Какое свойство описывает индекс выделенного элемента в ComboBox?
2. добавить элемент в в ComboBox?
3. Как получить значение, записанное в в ComboBox?
4. Как закрепить панель инструментов?

Лабораторная работа № 10

Математические функции

Цель работы:

Закрепить навыки работы с основными компонентами Visual C#. Ознакомиться с библиотекой Math.dll.

Теоретические сведения

Как и в большинстве языков программирования, в языке C# имеется большой набор математических функций.
Все математические функции реализованы в виде статических методов в классе Math, который в свою очередь определён в области имён System.
Для того, чтобы в программе на C# использовать математические функции, необходимо подключать область имён System, а при вызове метода, реализующего ту или иную математическую функцию, явно указывать название класса Math.
Наиболее употребительные методы класса Math:

Abs(x)	Вычисляет модуль (абсолютное значение) числа x. Перегружен для всех числовых типов (int, double и т.д.)
Acos(x)	Функция арккосинуса. Значение аргумента должно находиться в диапазоне от -1 до +1
Asin(x)	Функция арксинуса. Значение аргумента должно находиться в диапазоне от -1 до +1
Atan(x)	Функция арктангенса
Cos(x)	Функция косинуса. Аргумент задается в радианах
Exp(x)	Вычисляет значение (экспоненциальная функция)
Log(x)	Возвращает значение натурального логарифма (ln x)
Log10(x)	Возвращает значение десятичного логарифма ( )
Max(a, b)	Возвращает максимум из двух чисел a и b
Min(a, b)	Возвращает минимум из двух чисел a и b
Pow(x, a)	Возвращает значение , то есть возводит число x в степень a
Sin(x)	Функция синуса. Угол задается в радианах
Sqrt(x)	Возвращает положительное значение квадратного корня

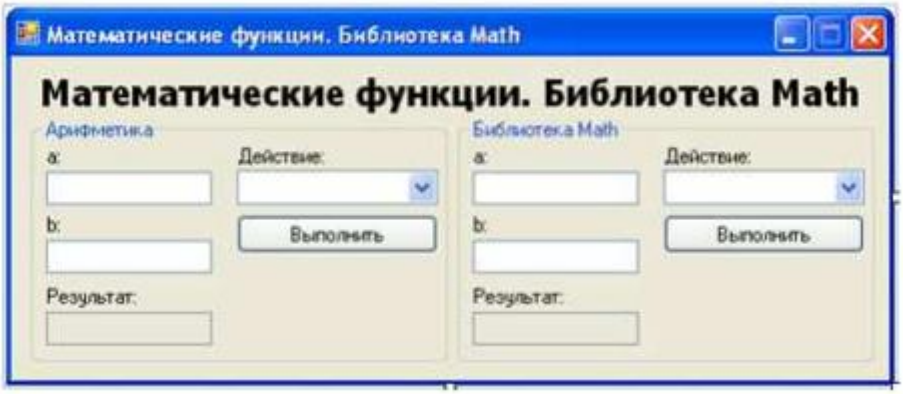
Tan(x)	Функция тангенса. Угол задается в радианах

Кроме методов, в классе имеется две константы:
Math.PI — равна числу = 3,14159265358979
Math.E — равна числу E = 2,71828182845905
При использовании математических функций необходимо не забывать указывать имя класса, а аргумент функции заключать в круглые скобки.
Примеры использования математических функций.

1)Вычислить
На C# это запишется так:
y = Math.Sin(x + Math.PI / 10);

2)Вычислить
В тексте программы на C# это будет выглядеть так:
y = Math.Sqrt(Math.Abs(x + 2))/(Math.Pow(x, 1.0 / 3) + 4);

Задание
Создайте программу с математическими функциями:



Описание:
1. В разделе Арифметика список Действие содержит опера- ции: сумма, разность, произведение, частное, остаток от деления.
2. В разделе Библиотека список Действие содержит операции: абсолютная величина, арккосинус, арктангенс, косинус, тангенс, экспонента, число Пи, десятичный логарифм, логарифм, округление, извлечение квадратного корня.
3. Программа должна отслеживать некорректный ввод данных: данных, не принадлежащих области определения функции, или ввод текста.

Порядок выполнения:
1. Изучите теоретический материал по теме лабораторной работы;
2. Создайте проект Windowx Form
3. Выполните действия по созданию форм приложения
4. В коде программы опишите логику приложения
5. Протестируйте программу
6. Оформите отчет

Форма отчетности:

Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;

4. Выводы по работе.

Основная литература

1. Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.) и предыдущие издания.

Контрольные вопросы для самопроверки

1. Какое свойство описывает индекс выделенного элемента в ComboBox?
2. добавить элемент в в ComboBox?
3. Как получить значение, записанное в в ComboBox?
4. Как закрепить панель инструментов?

Лабораторная работа № 11

Панель инструментов

Цель работы:

Приобрести навыки создания различных видов программного меню, осуществления переходов между формами, ознакомление с дочерними формами..

Теоретические сведения

Создание дочерней формы

Дочерняя форма (MDI-форма) – это форма, которая существует в пределах другой формы (родительской) и до тех пор, пока не будет закрыта или сама дочерняя форма или родительская. Открыть дочернюю форму без родительской формы невозможно. Переместить дочернюю форму за пределы родительской формы невозможно.

Чтобы создать дочернюю форму, выполните следующие действия:

1. Выберите форму, которая должна стать родительской.
2. В окне Свойства этой формы задайте свойству IsMdiContainer значение true.
3. Создайте новую форму.
4. Вставьте в обработчик события открытия новой формы код:

```
private void button1_Click(object sender, EventArgs e)
 {
 // Создание переменной формы
 Form3 frm_child = new Form3();
// Присваивание форме в качестве родителя текущую форму
frm_child.MdiParent = this;
 // Открытие формы frm_child.Show();
}
```

Примечание. Чтобы предотвратить повторное создание дочерней формы при уже открытой, вышепрописанный код следует изменить:

1. Создайте глобальную переменную дочерней формы
2. Вставьте в обработчик события открытия новой формы код:

```
private void button1_Click(object sender, EventArgs e)
{
if (glob.frm_child == null)
{
glob.frm_child = new Form3();
 glob.frm_child.MdiParent = this;
glob.frm_child.Show();
 }
 }
```

Создать обработчик события закрытия дочерней формы FormClosed:

```
private void Form3_FormClosed(object sender, FormClosedEventArgs e)
{
glob.frm_child = null;
 }
```

В этой функции глобальной переменной дочерней формы при- сваивается значение null, фактически освобождая память.

Настройка формы

Все действия по изменению внешнего вида формы производятся с помощью команд, отображаемых в окне Свойства в рабочей области проекта.

Для отображения окна Свойства необходимо выбрать в контекстном меню к форме команду Свойства. В результате открывается окно со свойствами формы. В верхней строке окна Свойства жирным шрифтом прописывается имя компонента, свойства которого отображаются в этом окне. Справа от имени отображается класс, которому принадлежит данный компонент.

Внимание! В конструкторе форм можно изменять размеры формы, но нельзя менять ее местоположение. Чтобы отобразить форму Windows Forms, в конструкторе дважды щелкните форму в Обозревателе решений.

Просмотр кода для формы

Перейти к коду для формы можно одним из следующих способов:

- В Обозревателе решений выберите форму и нажмите кнопку Просмотреть код
- В Обозревателе решений в контекстном меню к форме выберите пункт Перейти к коду.
- Если фокус находится в конструкторе, нажмите клавишу F7 для переключения в режим Редактор кода.

Примечание. Двойной щелчок формы или ее элемента управления в конструкторе также переключает в режим Редактора кода, но при этом добавляет обработчик событий по умолчанию для этого элемента управления. Например, двойной щелчок элемента управления Button приводит к отображению Редактора кода и добавляет обработчик событий Button_Click

Создание глобальной переменной класса Form

Пусть имеется форма Form1. Обратиться программно в коде непосредственно к Form1 в C# нельзя.

Form1 – это класс. Для обращения к форме необходимо завести переменную этого класса.

Создать глобальную переменную для формы можно несколькими способами.

Способ 1. Создать глобальную переменную класса Form, а далее при инициализации прописать ее принадлежность классу Form1. Чтобы создать глобальную переменную формы этим способом, необходимо в пространстве решения в любом месте после описания формы прописать код public class имя_класса

```
{
 public static Form имя_переменной;
}
```

Пример:

```
namespace Metodichka
{
……. public class glob
{
 public static Form frm;
}

}
```

Провести инициализацию переменной: glob.frm = new Form1().

Способ 2. Создать сразу переменную заданного класса: public class имя_класса

```
{
 public static Form имя_перем = new имя_класса_формы();
}
```

Пример:

```
public class glob
{
 public static Form frm=new Form1();
}
```

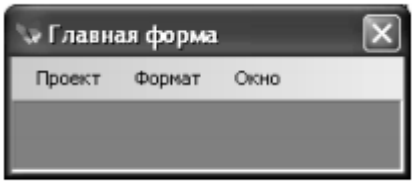
Переименование формы

Для переименования формы необходимо изменить значение свойства Text в окне Свойства. Программным путем переименование формы можно произвести с помощью команды: Имя_переменной_формы.Text = "Новое название"; Пример: glob.frm.Text = "Моя первая форма"; Примечание. Исключение составляет начальная форма. Чтобы переименовать программным способом начальную форму, необходимо переменную этой формы создать в файле program.cs, например: namespace Metodichka

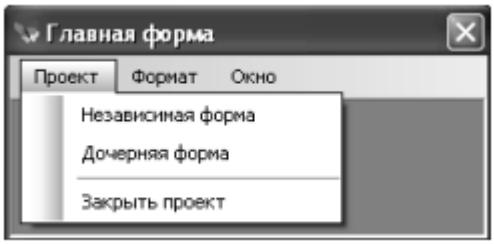
```
{
 // Создаем глобальную переменную public static class glob
{
 public static Form frm1;
}
 static class Program
{
 ….. static void Main()
{
 Application.EnableVisualStyles(); // инициализируем переменную glob.frm1 = new Form1(); 50
// запускаем проект Application.Run(glob.frm1);
} } }
```

Задание

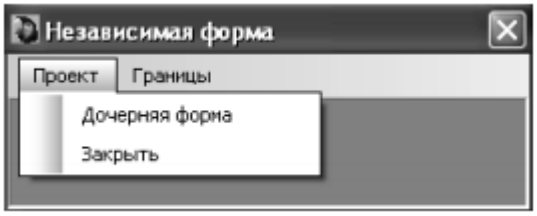
Создать проект из четырех форм: главная форма и дочерняя к ней, независимая форма и дочерняя к ней. Главная форма Главная форма должна иметь свою иконку, не содержать кнопки сворачивания и разворачивания окна, иметь жесткие границы.
В главной форме создать программное меню с пунктами Проект, Формат и Окно:



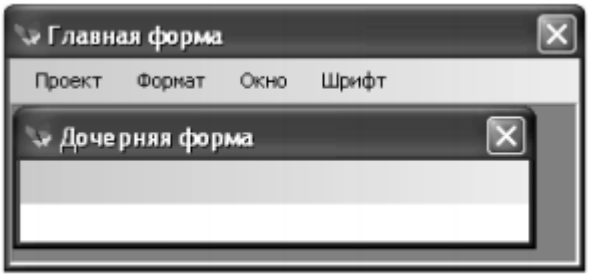
Пункт Проект главной формы Пункт
Проект должен содержать разделитель и команды: Не-зависимая форма, Дочерняя форма, Закрыть проект



Действия при выборе команд пункта Проект:
1. Независимая форма – открытие независимого окна со своим программным меню:



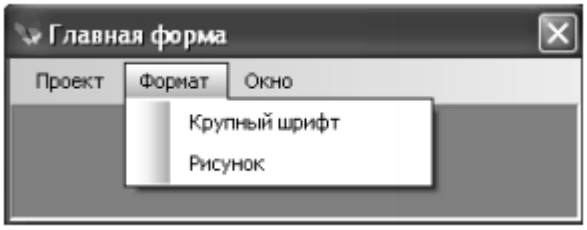
2. Дочерняя форма – открытие дочернего окна:



3. Закрыть проект – закрытие приложения.

Пункт Формат главной формы Пункт Формат должен содержать команды Крупный шрифт и Рисунок.

Действия при выборе команд пункта Формат:



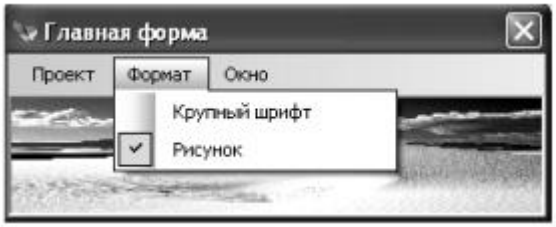
1. При выборе команды Крупный шрифт:

1) должен изменяться шрифт меню главной формы;

2) рядом с командой Крупный шрифт должен появляться сим- вол выбора команды ( );

3) при повторном выборе команды Крупный шрифт символ должен исчезать, а шрифт становится первоначальным.

2. При выборе команды Рисунок:

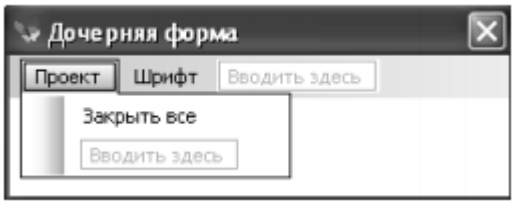


1) в качестве фона должен появляться рисунок, прописанный в коде программы;

2) рядом с командой Рисунок должен появляться символ вы- бора команды ( );

3) при повторном выборе команды Рисунок символ дол- жен исчезать, а фон становится первоначальным. Независимая форма Независимая форма должна иметь свою иконку и содержать свое программное меню с пунктами Проект и Границы. Пункт Проект независимой формы Пункт Проект должен содержать команды: Дочерняя форма и Закрыть:

Действия при выборе команд пункта Проект:



1. Дочерняя форма – открытие дочернего окна;

2. Закрыть – закрытие независимого окна. Пункт Границы независимой формы Пункт Границы должен содержать команды: Обычные, Отсутствуют и 3D: При открытии формы по умолчанию устанавливается тип гра- ниц Обычные и возле этого пункта меню должен стоять символ выбора .

Действия при выборе команд пункта Границы:

При выборе одной из команд пункта Границы:

1) должен изменяться стиль границ независимой формы;

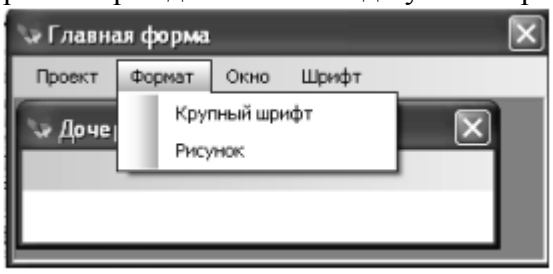
2) рядом с выбранным стилем границ должен появляться символ выбора команды ( ).

Дочерняя форма к главной форме Дочерняя форма должна содержать собственное программное меню, встраиваемое в меню главной формы. Программное меню дочерней формы должно состоять из пунктов: Проект и Шрифт. При открытии из главной формы дочерней формы команда Проект главной формы должна заменяться командой Проект дочерней формы. Команда Шрифт должна добавляться в меню главной формы после команды Окно. При открытии дочерней формы из главной формы название дочерней формы должно отображаться в меню Окно главной формы

Меню Формат должно остаться без изменений и содержать команды: Крупный шрифт и Рисунок. Пункт Проект дочерней формы к главной форме Пункт Проект должен содержать одну команду Закрыть все, при выборе которой закрывается приложение. Пункт Шрифт дочерней формы к главной форме Пункт Шрифт должен содержать команды Обычный и курсив. При открытии дочерней формы по умолчанию устанавливается стиль шрифта Обычный и возле этого пункта меню должен стоять символ выбора .

Действия при выборе команд пункта Шрифт:

При выборе одной из команд пункта Шрифт:



1) должен изменяться стиль шрифта программного меню. При этом при изменении стиля шрифта должен сохраняться его размер, установленный в пункте Формат. Соответственно, при изменении размера шрифта должен сохраняться стиль шрифта, выбранный в пункте Шрифт.

2) рядом с выбранным стилем границ должен появляться сим- вол выбора команды ( ).

Дочерняя форма к независимой форме Дочерняя форма должна иметь такую же, как у независимой формы, иконку и содержать собственное программное меню, встраиваемое в меню независимой формы. Программное меню до- черней формы должно состоять из пунктов: Проект и Фон. При открытии дочерней формы к независимой форме программное меню дочерней формы должно полностью заменяться программным меню дочерней формы. Пункт Проект дочерней формы к независимой форме Пункт Проект должен содержать одну команду Закрыть, при выборе которой закрывается независимая форма

Пункт Фон дочерней формы к независимой форме Пункт Фон должен содержать команды Отсутствует, Ветер и Восхождение. При открытии дочерней формы по умолчанию фоновый рисунок отсутствует, поэтому рядом с командой Отсутствует должен стоять символ выбора .

Действия при выборе команд пункта Фон:

При выборе одной из команд пункта Фон:

1) должен изменяться фон дочерней формы;

2) рядом с выбранным фоном должен появляться символ вы- бора команды ( ).

Порядок выполнения:

1. Изучите теоретический материал по теме лабораторной работы;
2. Создайте проект Windowx Form
3. Выполните действия по созданию форм приложения
4. В коде программы опишите логику приложения
5. Протестируйте программу
6. Оформите отчет

Форма отчетности:

Отчет по работе содержит:

1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.) и предыдущие издания.

Контрольные вопросы для самопроверки

1. Что такое дочерняя форма?
2. Чем независимая форма отличается от дочерней?
3. Как программно обрабатывается меню?
4. Какой командой компилируется программа в gcc?

Лабораторная работа № 12

Таблицы

Цель работы:

Приобрести навыки создания панели инструментов, вкладок и создания глобальных методов. Краткие теоретические сведения

Теоретические сведения

DefaultExt: устанавливает расширение файла, которое добавляется по умолчанию, если пользователь ввел имя файла без расширения

AddExtension: при значении true добавляет к имени файла расширение при его отсуствии. Расширение берется из свойства DefaultExt или Filter

CheckFileExists: если имеет значение true, то проверяет существование файла с указанным именем

CheckPathExists: если имеет значение true, то проверяет существование пути к файлу с указанным именем

FileName: возвращает полное имя файла, выбранного в диалоговом окне

Filter: задает фильтр файлов, благодаря чему в диалоговом окне можно отфильтровать файлы по расширению. Фильтр задается в следующем формате Название_файлов|*.расширение. Например, Текстовые файлы(*.txt)|*.txt. Можно задать сразу несколько фильтров, для этого они разделяются вертикальной линией |. Например, Bitmap files (*.bmp)|*.bmp|Image files (*.jpg)|*.jpg

InitialDirectory: устанавливает каталог, который отображается при первом вызове окна

Title: заголовок диалогового окна

Отдельно у класса SaveFileDialog можно еще выделить пару свойств:

CreatePrompt: при значении true в случае, если указан не существующий файл, то будет отображаться сообщение о его создании

OverwritePrompt: при значении true в случае, если указан существующий файл, то будет отображаться сообщение о том, что файл будет перезаписан

Чтобы отобразить диалоговое окно, надо вызвать метод ShowDialog().

Рассмотрим оба диалоговых окна на примере. Добавим на форму текстовое поле textBox1 и две кнопки button1 и button2. Также перетащим с панели инструментов компоненты OpenFileDialog и SaveFileDialog

Задание

6. Написать функции для преобразования декартовых координат в полярные и обратно. Продемонстрировать их вызов в программе.
7. Напишите функцию, которая находит площадь прямоугольника.
   Напишите программу, которая использует эту функцию. Из 10 прямоугольников программа находит тот, у которого площадь наибольшая.
8. Написать функцию Ugol для вычисления угла в треугольнике. Напишите программу, которая определяет все углы в треугольнике, вершины которого вводит пользователь.

9. Напишите функцию, которая вычисляет расстояние между двумя точками на плоскости. Напишите программу, которая использует эту функцию. Программа определяет для N введенных точек две наиболее близкие.
10. Напишите функцию, которая находит сумму цифр целого числа N ($N<2^{15}$). Напишите программу, которая использует эту функцию. Программа обрабатывает массив из K целых чисел и печатает для каждого из них сумму цифр.

Порядок выполнения:

1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с

Дополнительная литература

Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки
1. Что представляет собой поток (stream)?
2. Сформулируйте основные принципы работы с массивами.
3. Сформулируйте основные принципы работы с двоичным потоком.
4. Перечислите классы для работы с таблицами

Лабораторная работа № 13

Файлы

Цель работы:
Приобрести навыки работы с текстовыми файлами и диалоговыми окнами открытия и сохранения информации.

Теоретические сведения
Окна открытия и сохранения файла представлены классами OpenFileDialog и SaveFileDialog. Они имеют во многом схожую функциональность, поэтому рассмотрим их вместе.
OpenFileDialog и SaveFileDialog имеют ряд общих свойств, среди которых можно выделить следующие:
DefaultExt: устанавливает расширение файла, которое добавляется по умолчанию, если пользователь ввел имя файла без расширения
AddExtension: при значении true добавляет к имени файла расширение при его отсуствии. Расширение берется из свойства DefaultExt или Filter
CheckFileExists: если имеет значение true, то проверяет существование файла с указанным именем
CheckPathExists: если имеет значение true, то проверяет существование пути к файлу с указанным именем
FileName: возвращает полное имя файла, выбранного в диалоговом окне

Filter: задает фильтр файлов, благодаря чему в диалоговом окне можно отфильтровать файлы по расширению. Фильтр задается в следующем формате Название_файлов|*.расширение. Например, Текстовые файлы(*.txt)|*.txt. Можно задать сразу несколько фильтров, для этого они разделяются вертикальной линией |. Например, Bitmap files (*.bmp)|*.bmp|Image files (*.jpg)|*.jpg
InitialDirectory: устанавливает каталог, который отображается при первом вызове окна
Title: заголовок диалогового окна
Отдельно у класса SaveFileDialog можно еще выделить пару свойств:
CreatePrompt: при значении true в случае, если указан не существующий файл, то будет отображаться сообщение о его создании
OverwritePrompt: при значении true в случае, если указан существующий файл, то будет отображаться сообщение о том, что файл будет перезаписан
Чтобы отобразить диалоговое окно, надо вызвать метод ShowDialog().
Рассмотрим оба диалоговых окна на примере. Добавим на форму текстовое поле textBox1 и две кнопки button1 и button2. Также перетащим с панели инструментов компоненты OpenFileDialog и SaveFileDialog. После добавления они отобразятся внизу дизайнера формы. В итоге форма будет выглядеть примерно так:

Теперь изменим код формы:

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        button1.Click += button1_Click;
        button2.Click += button2_Click;
        openFileDialog1.Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*";
        saveFileDialog1.Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*";
    }
    // сохранение файла
    void button2_Click(object sender, EventArgs e)
    {
        if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)
            return;
        // получаем выбранный файл
        string filename = saveFileDialog1.FileName;
        // сохраняем текст в файл
        System.IO.File.WriteAllText(filename, textBox1.Text);
        MessageBox.Show("Файл сохранен");
    }
    // открытие файла
    void button1_Click(object sender, EventArgs e)
    {
        if (openFileDialog1.ShowDialog() == DialogResult.Cancel)
            return;
        // получаем выбранный файл
        string filename = openFileDialog1.FileName;
        // читаем файл в строку
        string fileText = System.IO.File.ReadAllText(filename);
        textBox1.Text = fileText;
        MessageBox.Show("Файл открыт");
    }
}
```

По нажатию на первую кнопку будет открываться окно открытия файла. После выбора файла он будет считываться, а его текст будет отображаться в текстовом поле. Клик на вторую кнопку отобразит окно для сохранения файла, в котором надо установить его название. И после этого произойдет сохранение текста из текстового поля в файл.
Задание:
По команде «Сохранить» открывается стандартное окно Windows по работе с файловой системой. Данные записываются в спе- циально отведенный для этого каталог в текстовый файл с расши- рением *.myp (my program), например, в следующем виде:

Дано:			
1	0	2	3
–5	7	0	1
2	9	–3	5
1	1	6	11

Результат:			
3	0	2	1
1	7	0	–5
5	9	–3	2
11	1	6	1

По команде «Открыть» открывается стандартное окно Windows по работе с файловой системой. Первая матрица (дано:) счи- тывается из выбранного файла и отображается на второй форме

Порядок выполнения:
1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.) и предыдущие издания.

Дополнительная литература

2. Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки
1. Что представляет собой поток (stream)?
2. Сформулируйте основные принципы работы с байтовым потоком.
3. Сформулируйте основные принципы работы с символьным потоком.
4. Сформулируйте основные принципы работы с двоичным потоком.
5. Перечислите классы для работы с каталогами и файлами.

Лабораторная работа № 14

Отображение HTML-файлов

Цель работы:
Приобрести навыки работы с html_страницами.

Теоретические сведения
В Си-шарп есть пространство имен System.IO, в котором реализованы все необходимые нам классы для работы с файлами. Чтобы подключить это пространство имен, необходимо в самом начале программы добавить строку using System.IO. Для использования кодировок

еще добавим пространство using System.Text;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
```

Как создать файл?
Для создания пустого файла, в классе File есть метод Create(). Он принимает один аргумент – путь. Ниже приведен пример создания пустого текстового файла new_file.txt на диске D:

```
static void Main(string[] args)
{
  File.Create("D:\\new_file.txt");
}
```

Если файл с таким именем уже существует, он будет переписан на новый пустой файл.
Метод WriteAllText() создает новый файл (если такого нет), либо открывает существующий и записывает текст, заменяя всё, что было в файле:

```
static void Main(string[] args)
{
  File. WriteAllText("D:\\new_file.txt", "текст");
}
```

Метод AppendAllText() работает, как и метод WriteAllText() за исключением того, что новый текст дописывается в конец файла, а не переписывает всё что было в файле:

```
static void Main(string[] args)
{
  File.AppendAllText("D:\\new_file.txt", "текст метода AppendAllText ()"); //допишет текст в конец файла
}
```

Как удалить файл?
Метод Delete() удаляет файл по указаному пути:

```
static void Main(string[] args)
{
  File.Delete("d:\\test.txt"); //удаление файла
}
```

Кроме того, чтобы читать/записывать данные в файл с Си-шарп можно использовать потоки. Поток – это абстрактное представление данных (в байтах), которое облегчает работу с ними. В качестве источника данных может быть файл, устройство ввода-вывода, принтер.

Класс Stream является абстрактным базовым классом для всех потоковых классов в Си-шарп. Для работы с файлами нам понадобится класс FileStream(файловый поток).

FileStream - представляет поток, который позволяет выполнять операции чтения/записи в файл.

```
static void Main(string[] args)
{
  FileStream file = new FileStream("d:\\test.txt", FileMode.Open
```

```
, FileAccess.Read); //открывает файл только на чтение
}
```

Режимы открытия FileMode:

- Append – открывает файл (если существует) и переводит указатель в конец файла (данные будут дописываться в конец), или создает новый файл. Данный режим возможен только при режиме доступа FileAccess.Write.
- Create - создает новый файл(если существует – заменяет)
- CreateNew – создает новый файл (если существует – генерируется исключение)
- Open - открывает файл (если не существует – генерируется исключение)
- OpenOrCreate – открывает файл, либо создает новый, если его не существует
- Truncate – открывает файл, но все данные внутри файла затирает (если файла не существует – генерируется исключение)

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\file1.txt", FileMode.CreateNew); //создание нового файла
  FileStream file2 = new FileStream("d:\\file2.txt", FileMode.Open); //открытие существующего файла
  FileStream file3 = new FileStream("d:\\file3.txt", FileMode.Append); //открытие файла на дозапись в конец файла
}
```

Режим доступа FileAccess:

- Read – открытие файла только на чтение. При попытке записи генерируется исключение
- Write - открытие файла только на запись. При попытке чтения генерируется исключение
- ReadWrite - открытие файла на чтение и запись.

Чтение из файла

Для чтения данных из потока нам понадобится класс StreamReader. В нем реализовано множество методов для удобного считывания данных. Ниже приведена программа, которая выводит содержимое файла на экран:

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\test.txt", FileMode.Open); //создаем файловый поток
  StreamReader reader = new StreamReader(file1); // создаем «потоковый читатель» и связываем его с файловым потоком
  Console.WriteLine(reader.ReadToEnd()); //считываем все данные с потока и выводим на экран
  reader.Close(); //закрываем поток
  Console.ReadLine();
}
```

Метод ReadToEnd() считывает все данные из файла. ReadLine() – считывает одну строку (указатель потока при этом переходит на новую строку, и при следующем вызове метода будет считана следующая строка).

Свойство EndOfStream указывает, находится ли текущая позиция в потоке в конце потока (достигнут ли конец файла). Возвращает true или false.

Запись в файл

Для записи данных в поток используется класс StreamWriter. Пример записи в файл:

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\test.txt", FileMode.Create); //создаем файловый поток
 StreamWriter writer = new StreamWriter(file1); //создаем «потоковый писатель» и связываем его с файловым потоком
  writer.Write("текст"); //записываем в файл
  writer.Close(); //закрываем поток. Не закрыв поток, в файл ничего не запишется
}
```

Метод WriteLine() записывает в файл построчно (то же самое, что и простая запись с помощью Write(), только в конце добавляется новая строка).

Нужно всегда помнить, что после работы с потоком, его нужно закрыть (освободить ресурсы), использовав метод Close().

Кодировка, в которой будут считываться/записываться данные указывается при создании StreamReader/StreamWriter:

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\test.txt", FileMode.Open);
  StreamReader reader = new StreamReader(file1, Encoding.Unicode);
  StreamWriter writer = new StreamWriter(file1, Encoding.UTF8);
}
```

Кроме того, при использовании StreamReader и StreamWriter можно не создавать отдельно файловый поток FileStream, а сделать это сразу при создании StreamReader/StreamWriter:

```
static void Main(string[] args)
{
  StreamWriter writer = new StreamWriter("d:\\test.txt"); //указываем путь к файлу, а не поток
  writer.WriteLine("текст");
  writer.Close();
}
```

Как создать папку?

С помощью статического метода CreateDirectory() класса Directory:

```
static void Main(string[] args)
{
 Directory.CreateDirectory("d:\\new_folder");
}
```

Как удалить папку?
Для удаления папок используется метод Delete():

```
static void Main(string[] args)
{
  Directory.Delete("d:\\new_folder"); //удаление пустой папки
}
```

Если папка не пустая, необходимо указать параметр рекурсивного удаления - true:

```
static void Main(string[] args)
{
  Directory.Delete("d:\\new_folder", true); //удаление папки, и всего, что внутри
}
```

Задание:

1. В лабораторной работе № 16 в программное меню добавить опцию «Справка» с пунктами «Матрицы», «О программе», «Клави- атура».
● При выборе команды «Матрицы» открывается окно с HTML-документом. HTML-документ должен быть создан само- стоятельно, содержать информацию о матрицах, о способе обраще- ния к элементу матрицы, описание свойства сложения матриц, пример сложения матриц. Текст должен быть отформатирован, иметь заголовки, разные стили и начертания шрифта.
● При выборе команды «О программе» открывается окно с HTML-документом. HTML-документ должен быть создан само- стоятельно, содержать информацию о программе и ее авторах.
● При выборе команды «Клавиатура» открывается окно с HTML-документом. HTML-документ должен быть создан само- стоятельно, содержать информацию о событиях клавиатуры из пункта
2. 2. Создать процедуры обработки событий нажатия клавиш клавиатуры:
● Enter – сложение матриц;
● Esc – очистка матриц.
3. Создать строку состояния, в которой отобразить время те- кущее, дату, подсказку при наведении мышки на объект.

Порядок выполнения:
1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.

Дополнительная литература

2. Павловская, Т. А. С/С++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки
1. Что представляет собой поток (stream)?
2. Сформулируйте основные принципы работы с байтовым потоком.
3. Сформулируйте основные принципы работы с символьным потоком.
4. Сформулируйте основные принципы работы с двоичным потоком.
5. Перечислите классы для работы с каталогами и файлами.

Лабораторная работа № 15

Бестиповые указатели

Цель работы : Приобрести навыки работы с динамическими переменными.

Теоретические сведения

В языке C# указатели очень редко используются, однако в некоторых случаях можно прибегать к ним для оптимизации приложений. Код, применяющий указатели, еще называют небезопасным кодом. Однако это не значит, что он представляет какую-то опасность. Просто при работе с ним все действия по использованию памяти, в том числе по ее очистке, ложится целиком на нас, а не на среду CLR. И с точки зрения CLR такой код не безопасен, так как среда не может проверить данный код, поэтому повышается вероятность различного рода ошибок.

Чтобы использовать небезопасный код в C#, надо первым делом указать проекту, что он будет работать с небезопасным кодом. Для этого надо установить в настройках проекта соответствующий флаг - в меню Project (Проект) найти Свойства проекта. Затем в меню Build установить флажокAllow unsafe code (Разрешить небезопасный код):

Теперь мы можем приступать к работе с небезопасным кодом и указателями.

Ключевое слово unsafe

Блок кода или метод, в котором используются указатели, помечается ключевым словом unsafe:

```
1    static void Main(string[] args)
2    {
3        // блок кода, использующий указатели
4        unsafe
5        {
6
7        }
8    }
```

Метод, использующий указатели:

```
1    unsafe private static void PointerMethod()
2    {
3
4    }
```

Также с помощью unsafe можно объявлять структуры:

```
1    unsafe struct State
2    {
3
4    }
```

Операции * и &

Ключевой при работе с указателями является операция *, которую еще называют операцией разыменовывания. Операция разыменовывания позволяет получить или установить значение по адресу, на который указывает указатель. Для получения адреса переменной применяется операция &:

```
1     static void Main(string[] args)
2     {
3         unsafe
4         {
5             int* x; // определение указателя
6             int y = 10; // определяем переменную
7
8             x = &y; // указатель x теперь указывает на адрес переменной y
9             Console.WriteLine(*x); // 10

              y = y + 20;
```

```
            Console.WriteLine(*x);// 30

            *x = 50;
            Console.WriteLine(y); // переменная y=50
        }
        Console.ReadLine();
    }
```

При объявлении указателя указываем тип int* x; - в данном случае объявляется указатель на целое число. Но кроме типа int можно использовать и другие: sbyte, byte, short, ushort, int, uint, long, ulong, char, float, double, decimal или bool. Также можно объявлять указатели на типы enum, структуры и другие указатели.

Выражение x = &y; позволяет нам получить адрес переменной y и установить на него указатель x. До этого указатель x не на что не указывал.

После этого все операции с y будут влиять на значение, получаемое через указатель x и наоборот, так как они указывают на одну и ту же область в памяти.

Для получения значения, которое хранится в области памяти, на которую указывает указатель x, используется выражение *x.

Получение адреса

Используя преобразование указателя к целочисленному типу, можно получить адрес памяти, на который указывает указатель:

```
1  int* x; // определение указателя
2  int y = 10; // определяем переменную
3
4  x = &y; // указатель x теперь указывает на адрес переменной y
5
6  // получим адрес переменной y
7  uint addr = (uint)x;
8  Console.WriteLine("Адрес переменной y: {0}", addr);
```

Так как значение адреса - это целое число, а на 32-разрядных системах диапазон адресов 0 до 4 000 000 000, то для получения адреса используется преобразование в тип uint, long или ulong. Соответственно на 64-разрядных системах диапазон доступных адресов гораздо больше, поэтому в данном случае лучше использовать ulong, чтобы избежать ошибки переполнения.

Операции с указателями

Кроме операции разыменовывания к указателям применимы еще и некоторые арифметические операции(+, ++, -, --, +=, -=) и преобразования. Например, мы можем преобразовать число в указатель:

```
int* x; // определение указателя
int y = 10; // определяем переменную
x = &y; // указатель x теперь указывает на адрес переменной y

// получим адрес переменной y
uint addr = (uint)x;
Console.WriteLine("Адрес переменной y: {0}", addr);

byte* bytePointer = (byte*)(addr+4); // получить указатель на следующий байт после addr
Console.WriteLine("Значение byte по адресу {0}: {1}", addr+4, *bytePointer);

// обратная операция
uint oldAddr = (uint)bytePointer - 4; // вычитаем четыре байта, так как bytePointer - указатель на байт
int* intPointer = (int*)oldAddr;
Console.WriteLine("Значение int по адресу {0}: {1}", oldAddr, *intPointer);
```

```
// преобразование в тип double
double* doublePointer = (double*)(addr + 4);
Console.WriteLine("Значение double по адресу {0}: {1}", addr + 4, *doublePointer);
```

Так как у нас x - указатель на объект int, который занимает 4 байта, то мы можем получить следующий за ним байт с помощью выражения byte* chp = (byte*)addr+4;. Теперь указатель bytePointer указывает на следующий байт. Равным образом мы можем создать и другой указательdouble* doublePointer = (double*)addr + 4;, только этот указатель уже будет указывать на следующие 8 байт, так как тип double занимает 8 байт.

Чтобы обратно получить исходный адрес, вызываем выражение bytePointer - 4. Здесь bytePointer - это указатель, а не число, и операции вычитания и сложения будут происходить в соответствии с правилами арифметики указателей. Например:

```
1  char* charPointer = (char*)123400;
2  charPointer += 4; // 123408
3  Console.WriteLine("Адрес {0}", (uint)charPointer);
```

Хотя мы к указателю прибавляем число 4, но итоговый адрес увеличится на 8, так как размер объекта char - 2 байта, а 2*4=8. Подобным образом действует сложение с другими типа указателей:

```
1  double* doublePointer = (double*)123000;
2  doublePointer = doublePointer+3; // 123024
3  Console.WriteLine("Адрес {0}", (uint)doublePointer);
```

Аналогично работает вычитание: doublePointer -=2 установит в указателе doublePointer в качестве адреса число 123008

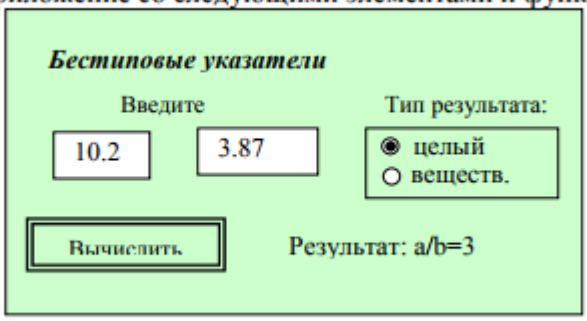
Указатель на другой указатель

Объявление и использование указателя на указатель:

```
        static void Main(string[] args)
        {
            unsafe
1           {
2               int* x; // определение указателя
3               int y = 10; // определяем переменную
4
5               x = &y; // указатель x теперь указывает на адрес переменной y
6               int** z = &x; // указатель z теперь указывает на адрес, который указывает и указатель x
7               **z = **z + 40; // изменение указателя z повлечет изменение переменной y
8               Console.WriteLine(y); // переменная y=50
9               Console.WriteLine(**z); // переменная **z=50
            }
            Console.ReadLine();
        }
```

Задание:

Создать приложение со следующими элементами и функциями:



где *a* и *b* – бестиповые указатели.

Порядок выполнения:

1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.

3. Протестировать приложение.

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с

Дополнительная литература

2. Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки
1. Что представляет собой указатель?
2. Почему код, использующий указатели считается небезопасным?.
3. Как записывается операция разыменования?
4. Какие операции можно осуществлять с указателями?
5. Как объявить указатель на указатель?

Лабораторная работа № 16

Связные списки

Цель работы :
Закрепить навыки работы с динамическими перемен- ными и приобрести навыки работы со связными списками..

Теоретические сведения

Связный список (Linked List) представляет набор связанных узлов, каждый из которых хранит собственно данные и ссылку на следующий узел. В реальной жизни связный список можно представить в виде поезда, каждый вагон которого может содержать некоторый груз или пассажиров и при этом может быть связан с другим вагоном.
Таким образом, если в массиве положение элементов определяется индексами, то в связном списке - указателями на следующий и (или) на предыдущий элемент.
Связные списки могут различаться. Есть односвязные списки, в которых каждый узел хранит указатель только на следующий узел. Есть двусвязные списки: в них каждый элемент хранит ссылку как на следующий элемент, так и на предыдущий. Есть кольцевые замкнутые списки. В данном случае мы рассмотрим создание односвязного списка.
Перед созданием списка нам надо определить класс узла, который будет представлять одиночный объект в списке:

```
1    public class Node<T>
2    {
3        public Node(T data)
4        {
5            Data = data;
6        }
7        public T Data { get; set; }
8        public Node<T> Next { get; set; }
9    }
```

Класс Node является обобщенным, поэтому может хранить данные любого типа. Для хранения данных предназначено свойство Data. Для ссылки на следующий узел определено свойство Next.
Далее определим сам класс списка:

```
using System.Collections;
using System.Collections.Generic;

namespace SimpleAlgorithmsApp
{
    public class LinkedList<T> : IEnumerable<T>  // односвязный список
    {
        Node<T> head; // головной/первый элемент
        Node<T> tail; // последний/хвостовой элемент
        int count;  // количество элементов в списке

        // добавление элемента
        public void Add(T data)
        {
            Node<T> node = new Node<T>(data);

            if (head == null)
                head = node;
            else
                tail.Next = node;
            tail = node;

            count++;
        }
        // удаление элемента
        public bool Remove(T data)
        {
            Node<T> current = head;
            Node<T> previous = null;
}
```

Разберем основные моменты. В зависимости от конкретных задач реализация списков может отличаться, но для всех реализаций характерны прежде всего два метода: добавление и удаление.

Но прежде чем выполнять различные операции с данными, в классе списка определяются три переменные:

```
1    Node<T> head; // головной/первый элемент
2    Node<T> tail; // последний/хвостовой элемент
3    int count;  // количество элементов в списке
```

Метод добавления:

```
public void Add(T data)
{
    Node<T> node = new Node<T>(data);

    if (head == null)
        head = node;
    else
        tail.Next = node;

    tail = node;
    count++;
}
```

Если у нас не установлена переменная head (то есть список пуст), то устанавливаем head и tail. После добавления первого элемента они будут указывать на один и тот же объект.

Если же в списке есть как минимум один элемент, то устанавливаем свойство tail.Next - теперь оно хранит ссылку на новый узел. И переустанавливаем tail - теперь она ссылается на новый узел.
Сложность данного метода составляет O(1). Графически это выглядит так:

Важно отметить наличие переменной tail, которая указывает на последний элемент. Ряд реализаций не используют подобную переменную и добавляют иным образом:

```
public void AddWithoutTail(T data)
{
   Node<T> node = new Node<T>(data);

   if (head == null)
   {
      head = node;
   }
   else
   {
      Node<T> current = head;
      // ищем последний элемент
      while (current.Next!=null)
      {
         current = current.Next;
      }
      //устанавливаем последний элемент
      current.Next = node;
   }
   count++;
}
```

Данный способ вполне рабочий и нередко встречается, однако необходимость перебора элементов для нахождения последнего увеличивает время на поиск и сложность алгоритма. Она равна O(n).
Особняком стоит метод добавления в начало списка, где нам достаточно переустановить ссылку на головной элемент:

```
public void AppendFirst(T data)
{
   Node<T> node = new Node<T>(data);
   node.Next = head;
   head = node;
   if (count == 0)
      tail = head;
   count++;
}
```

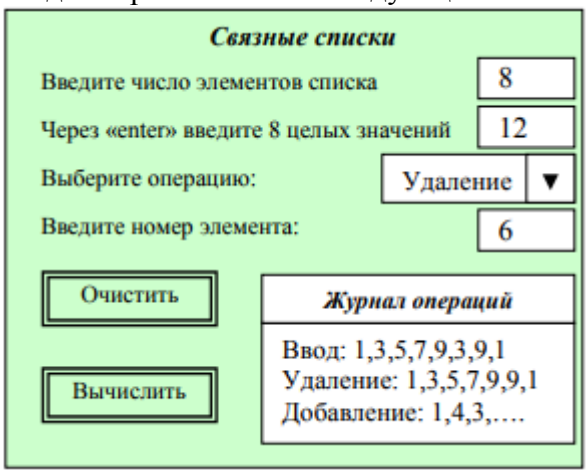
Удаление элемента

```
public bool Remove(T data)
{
   Node<T> current = head;
   Node<T> previous = null;

   while (current != null)
   {
      if (current.Data.Equals(data))
      {
         // Если узел в середине или в конце
         if (previous != null)
```

```
            {
                previous.Next = current.Next;
                if (current.Next == null)
                    tail = previous;
            }
            else
            {
                head = head.Next;

                if (head == null)
                    tail = null;
            }
            count--;
            return true;
        }
        previous = current;
        current = current.Next;
    }
    return false;
}
```

Алгоритм удаления элемента представляет следующую последовательность шагов:
Поиск элемента в списке путем перебора всех элементов
Установка свойства Next у предыдущего узла (по отношению к удаляемому) на следующий узел по отношению к удаляемому.
Для отслеживания предыдущего узла применяется переменная previous. Если элемент найден, и переменная previous равна null, то удаление идет сначала, и в этом случае происходит переустановка переменной head, то есть головного элемента.
Если же previous не равна null, то реализуются шаги выше описанного алгоритма.

Задание:
Создать приложение со следующими элементами и функциями:



Примечание. Кнопка «Вычислить» запускает выбранную опе- рацию; кнопка «Очистить» очищает все поля. Список операций содержит операции: «нет», «удалить», «вставить», «заменить», «найти»

Порядок выполнения:
1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:
Отчет по работе содержит:

1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.) и предыдущие издания.

Дополнительная литература

2. Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки

1. Что представляет собой указатель?
2. Почему код, использующий указатели считается небезопасным?.
3. Как записывается операция разыменования?
4. Какие операции можно осуществлять с указателями?
5. Как объявить указатель на указатель?

Лабораторная работа №17

Графика

Цель работы: Приобрести навыки работы с графическими объектами различного вида.

Теоретические сведения

Перед тем как рисовать линии и фигуры, отображать текст, выводить изображения и
Получение ссылки на объект Graphics из объекта PaintEventArgs в событии Paint
Объявите объект Graphics.
Присвойте переменной ссылку на объект Graphics, передаваемый как часть PaintEventArgs.
Вставьте код для рисования формы или элемента управления.
В следующем примере показано, как создавать ссылку на объект Graphics из объекта PaintEventArgs события Paint.

```
private void Form1_Paint(object sender,
  System.Windows.Forms.PaintEventArgs pe)
{
  // Declares the Graphics object and sets it to the Graphics object
  // supplied in the PaintEventArgs.
  Graphics g = pe.Graphics;
  // Insert code to paint the form here.
}
```

Метод CreateGraphics

Для получения ссылки на объект Graphics, который соответствует поверхности рисования формы или элемента управления, можно также использовать метод CreateGraphics этой формы или элемента управления.
Создание объекта Graphics с помощью метода CreateGraphics
Вызовите метод CreateGraphics формы или элемента управления, на котором необходимо отобразить графику.
C#
C++
VB

```
Graphics g;
// Sets g to a graphics object representing the drawing surface of the
```

```
// control or form g is a member of.
g = this.CreateGraphics();
```
Создание из объекта Image
Объект Graphics можно создать из любого объекта, производного от класса Image.
Создание объекта Graphics из объекта Image
Вызовите метод Graphics.FromImage переменной Image, из которой нужно создать объект Graphics.
В следующем примере показывается, как использовать объект Bitmap.
```
Bitmap myBitmap = new Bitmap(@"C:\Documents and
  Settings\Joe\Pics\myPic.bmp");
Graphics g = Graphics.FromImage(myBitmap);
```

Рисование фигур и изображений и управление ими
После создания объекта Graphics его можно использовать для рисования линий и фигур, отображения текста или изображения и управления ими. Ниже представлены основные объекты, используемые с объектом Graphics.
Класс Pen — служит для рисования линий, контуров и отрисовки других геометрических объектов.
Класс Brush — служит для заливки областей, например фигур, изображений или текста.
Класс Font — содержит описание фигур, которые должны использоваться при отрисовке текста.
Структура Color — содержит различные цвета.
Использование созданного объекта Graphics
Используйте перечисленные выше объекты, чтобы рисовать нужные изображения.

Задание:

Создать приложение с четырьмя вкладками для выполнения следующих задач: 1. Построение на одной координатной плоскости разными стилями рисования графиков функций на заданных интервалах. Наличие координатных осей, рисок и меток обязательно
Запрограммировать элементы динамической графики:
№ Элементы динамической графики
1 Небо, мерцающие звезды, луна и исходящее сияние от нее
2 Семицветик, середина которого через равные промежутки времени 1 t меняет цвет с color1 на color2 и обратно, а лепестки меняют цвет по кругу
3 Ракета, старт, ее полет по кривой, уменьшение размеров, след от ракеты
4 Движение волн (не меньше 6 волн), плывущая рыбка, небо, солнце
5 Движущийся автомобиль, препятствие, при нажатии клавиши автомобиль должен остановиться. Чем ближе к препятствию он остановится, тем больше очков 204

Порядок выполнения:
1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:
Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.) и предыдущие издания.

Дополнительная литература

2. Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки

1. Какие классы используются для рисования линий?
2. Какие классы нужны для отрисовки ?
3. Зачем используется объект Graphics?
4. Какие операции можно осуществлять с указателями?
5.Какие элементы управления Вы использовали для размещения изображения?

Лабораторная работа № 18

Диаграммы

Цель работы: Приобрести навыки построения диаграмм.

Теоретические сведения

Вызовите метод CreateGraphics элемента управления или формы, чтобы получить ссылку на объект Graphics, соответствующий поверхности рисования этой формы или элемента управления.Используйте этот подход, если необходимо рисовать на поверхности уже существующей формы или элемента управления.

Создайте объект Graphics из любого объекта, унаследованного от класса Image. Этот способ используется, когда требуется изменить существующее изображение.

Ниже все эти процедуры описаны более подробно.

PaintEventArgs в обработчике события Paint

При создании обработчика события PaintEventHandler для элементов управления или объекта PrintPage для PrintDocument, графический объект предоставляется как одно из свойств объекта PaintEventArgs или PrintPageEventArgs.

Получение ссылки на объект Graphics из объекта PaintEventArgs в событии Paint

Объявите объект Graphics.

Присвойте переменной ссылку на объект Graphics, передаваемый как часть PaintEventArgs.

Вставьте код для рисования формы или элемента управления.

В следующем примере показано, как создавать ссылку на объект Graphics из объекта PaintEventArgs события Paint.

```
private void Form1_Paint(object sender,
  System.Windows.Forms.PaintEventArgs pe)
{
  // Declares the Graphics object and sets it to the Graphics object
  // supplied in the PaintEventArgs.
  Graphics g = pe.Graphics;
  // Insert code to paint the form here.
}
```

Метод CreateGraphics

Для получения ссылки на объект Graphics, который соответствует поверхности рисования формы или элемента управления, можно также использовать метод CreateGraphics этой формы или элемента управления.

Создание объекта Graphics с помощью метода CreateGraphics

Вызовите метод CreateGraphics формы или элемента управления, на котором необходимо отобразить графику.

```
Graphics g;
// Sets g to a graphics object representing the drawing surface of the
```

```
// control or form g is a member of.
g = this.CreateGraphics();
```

Создание из объекта Image

Объект Graphics можно создать из любого объекта, производного от класса Image.

Создание объекта Graphics из объекта Image

Вызовите метод Graphics.FromImage переменной Image, из которой нужно создать объект Graphics.

В следующем примере показывается, как использовать объект Bitmap.

```
Bitmap myBitmap = new Bitmap(@"C:\Documents and
  Settings\Joe\Pics\myPic.bmp");
Graphics g = Graphics.FromImage(myBitmap);
```

Задание:

Создать с помощью датчика случайных чисел таблицу показателей среднего балла

В программе обязательно должно быть:

● Наличие на панели инструментов команд, позволяющих: – выводить график с общим средним баллом по математике и русскому языку; – выводить точечный график с общим средним баллом по математике и русскому; – выводить два графика (баллы по математике и по русскому) на одной диаграмме; – строить гистограмму общего среднего балла; – строить гистограмму по отдельным значениям по математике и русскому языку. Все диаграммы должны содержать заголовок, подписи осей и не иметь возможности изменять масштаб.

● Наличие цветовой панели для гистограммы и двух графиков с возможностью изменения цветов диаграммы.

● Наличие легенды для диаграммы с двумя графиками.

● Наличие сеток основной и вспомогательной для диаграммы с одним графиком.

● На диаграмме с общим средним баллом найти и подписать минимальное и максимальное значения разными цветами.

Порядок выполнения:

1. Изучить теоретический материал.
2. Написать приложение, реализующее поставленную задачу.
3. Протестировать приложение.

Форма отчетности:

Отчет по работе содержит:

1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Орлов, С. А. Теория и практика языков программирования : учебник для бакалавров и магистров / С. А. Орлов. - Санкт-Петербург: Питер, 2014. - 688 с.) и предыдущие издания.

Дополнительная литература

2. Павловская, Т. А. C/C++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2007. - 239 с.

Контрольные вопросы для самопроверки

1. Что представляет собой указатель?
2. Почему код, использующий указатели считается небезопасным?
3. Как записывается операция разыменования?
4. Какие операции можно осуществлять с указателями?

5. Как объявить указатель на указатель?

9.2. Методические указания по выполнению курсовой работы

Курсовая работа по дисциплине Языки и методы программирования представляет письменный отчет по самостоятельной работе обучающегося в рамках заданной темы.

Основными целями курсовой работы являются:

- закрепление практических умений в области разработки программного обеспечения;
- углубление теоретических и практических знаний по дисциплине;
- развитие способности находить нужную информацию и применять её для решения - поставленных задач;
- формирование навыков планирования, реализации, анализа и оценки собственной деятельности.

Задачи курсовой работы определяются исходя из индивидуального задания, полученного обучающимся.

В общем случае, задачи курсовой следующие:

- изучить литературу по выбранной теме;
- проанализировать имеющиеся на данный момент алгоритмы, методы средства решения поставленной задачи;
- разработать программное приложение, полученное в виде задания на разработку.
- при необходимости, разработать собственные методы и алгоритмы решения поставленной задачи.

Этапы работы над курсовой

1. Выбор технического задания на разработку и формулировка соответствующей ему темы;
2. Подбор литературных источников по теме.
3. Составление предварительного варианта плана.
4. Написание текста курсовой работы и разработка приложений.
5. Оформление курсовой работы
6. Составление доклада и разработка презентации .
7. Защита курсовой работы.

Требования к качеству текста

Текст курсовой работы должен быть логичным и последовательным. Расплывчатые и объемные рассуждения авторов произведений, используемых в работе, следует представить лаконично и обоснованно. Каждое предложение должно быть литературно обработано. Стиль изложения материала должен быть единым по всей работе. Орфографические, грамматические и стилистические ошибки недопустимы.

Не рекомендуется вести изложение от первого лица: «Я считаю», «Мне кажется», «Я получил(а)». В отдельных случаях возможно использование выражений типа: «По нашему мнению», «По мнению автора выпускной квалификационной работы». Однако более предпочтительными оказываются фразы «на основе выполненного анализа можно утверждать», «проведенные исследования подтвердили…».

Сокращения в виде аббревиатуры допускаются только при условии их первоначального разъяснения.

Заголовки глав и параграфов должны быть короткими и содержательными, они не должны повторять названия темы курсовой работы.

Содержание глав основной части должно точно соответствовать теме работы и полностью её раскрывать.

В целом текст работы должен продемонстрировать умение автора сжато, точно, логично и аргументировано излагать информацию.

Требования к содержанию работы

Титульный лист оформляется по единому образцу, принятому на выпускающей кафедре. Образец оформления титульного листа представлен ниже:

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БРАТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра математики

КУРСОВАЯ РАБОТА

по дисциплине
Языки и методы программирования

Тема: Случайные числа в ОС Linux

Выполнил:
студент гр. ПМиИ-07 _____ Ф.И.О. студента

Проверил:
ст. преподаватель каф. математики _____ Ф.И.О. преподавателя

Братск 2017

Содержание включает перечисление частей работы, начиная от введения и заканчивая приложениями, с указанием страницы начала каждой части. Названия частей курсовой должны в точности соответствовать содержанию

Введение

Во введении, объем которого 1- 1,5 страницы печатного текста, указывается актуальность темы, формулируются цели и задачи работы. Здесь же приводится аннотация глав работы.

Первая глава

Содержанием первой главы являются, как правило, основные теоретические вопросы по выбранной теме.

Возможные структурные элементы главы:

1. анализ рассматриваемой проблемы (математической или информационной);
2. место и роль данной темы в науке (математике, информатике);
3. обзор алгоритмов и методов, применяемых для решения задач, схожих с поставленной;
4. получение и представление собственных результатов (теоретических и практических);
5. предположения и выводы.

Вторая глава

Содержит

постановку задачи на разработку программы,

1. подходы к решению проблемы: алгоритмы, методы, средства разработки;
2. описание функционала программной разработки, её пользовательского интерфейса,
3. общую структуру проекта;
4. набор тестовых примеров и запланированные результаты их применения;
5. описание результатов запуска и тестирования программы со скриншотами.

Код вставляется фрагментами с обязательным текстовым комментарием.

Заключение - последовательное логически стройное изложение итогов и их соотношение с общей целью и конкретными задачами, поставленными и сформулированными во введении. В заключении концентрированно отражаются все выводы и положения, содержащиеся в работе. Объём заключения от 2/3 страницы.

Список литературы

Список использованной литературы должен быть выполнен в соответствии с ГОСТ 7.32.2001 «Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления» и правилами библиографического описания документов ГОСТ 7.1-2003 «Библиографическая запись. Библиографическое описание». ГОСТы имеются в библиотеке университета, где можно получить и консультацию по оформлению списков литературы. Кроме того, подробное оформление библиографического списка имеется на сайте библиотеки БрГУ.

Список обязательно должен быть пронумерован. Каждый источник упоминается в списке один раз, вне зависимости от того, как часто на него делается ссылка в тексте работы.

Наиболее удобным является алфавитное расположение материала, так как в этом случае произведения собираются в авторских комплексах. Произведения одного автора расставляются в списке по алфавиту заглавий.

Официальные документы ставятся в начале списка в определенном порядке: Конституции; Кодексы; Законы; Указы Президента; Постановление Правительства; другие нормативные акты (письма, приказы и т.д.). Внутри каждой группы документы располагаются в хронологическом порядке.

Литература на иностранных языках ставится в конце списка после литературы на русском языке, образуя дополнительный алфавитный ряд.

Для каждого документа предусмотрены следующие элементы библиографической

характеристики: фамилия автора, инициалы; название; подзаголовочные сведения (учебник, учебное пособие, словарь и т. д.); выходные сведения (место издания, издательство, год издания); количественная характеристика (общее количество страниц в книге) (Приложение 5).

Требования к оформлению курсовой работы

1) текст работы печатается на одной стороне листа белой бумаги *формата А4;*

2) *объем* – 25–30 страниц.

3) *поля:* левое – 3 см, правое – 1,0 см, верхнее - 2,0 см, нижнее – 2,0 см;

4) *шрифт* Times New Roman по всей работе;

4) *размер букв*: основной текст –14, в таблицах и приложениях допускается использовать кегль 12;

5) *цвет* шрифта – черный;

6) *интервал* между строками – 1,5, в таблицах допустим одинарный интервал;

7) в*ыравнивание* по ширине;

8) *автоматические переносы* обязательны;

9) абзацный отступ – 1,27 см.;

10) все страницы работы нумеруются, начиная с титульного листа и до последней страницы приложений, номер страницы на титульном листе не проставляют;

11) разрешается использовать компьютерные возможности акцентирования внимания на определенных частях текста, терминах и определениях, применяя курсив.

Введение, заключение, литература, названия глав, параграфов и приложений печатаются **жирным** шрифтом по *левому краю листа с абзацным отступом 1,27см*.

Слова «Содержание», «Введение», «Заключение», «Литература», названия глав и параграфов печатаются с прописной буквы, последующие буквы – строчные.

После названий глав и параграфов *точки не ставятся.* Не рекомендуется подчеркивать заголовки.

В заголовках *не допускается* перенос слов, сокращение слов и применение аббревиатур. Если заголовок состоит из двух предложений, их разделяют точкой, а после второго предложения точка не ставится.

Не допускается расположение заголовка на одной странице, а текста – на другой. Если заголовок размещается в нижней части страницы, то после него должно быть не менее трех строк текста. В противном случае, заголовок и текст переносятся на следующую страницу.

Главы и параграфы нумеруются арабскими цифрами. Принадлежность параграфа главе отмечается двумя цифрами, первая из которых – номер главы, а вторая (через точку) – номер параграфа. Например, «1.2 Информационные технологии в программировании» – второй параграф первой главы. Нумерация параграфов едина внутри главы. В каждой главе должно быть не менее двух параграфов.

Если имеется необходимость выделить части параграфа (не менее двух), их тоже нумеруют. К примеру, заголовок «1.2.1 Среда разработки Qt» представляет первую часть второго параграфа первой главы. И таких частей должно быть не менее двух.

Между заголовком главы и параграфа, между заголовком параграфа и последующим текстом – полуторный интервал. Каждая глава, введение, заключение, список литературы и приложения начинаются с новой страницы.

Рисунки, графики, схемы, таблицы могут иметь либо сквозную нумерацию по всей работе, либо нумерацию, как и параграфы, соответствующую данной главе.

При наличии в тексте списков перед каждой позицией следует ставить дефис (–), другие маркеры не допускаются, запись производится с абзацного отступа.

Если на перечисления в тексте делаются ссылки, то необходимо использовать строчную букву или арабские цифры, после которых ставится скобка (если пункты перечисления будут разделены точкой с запятой; в этом случае каждый пункт начинается со строчной буквы).

Если каждый пункт содержит в себе законченную мысль (одно или несколько предложений) в этом случае каждый пункт начинается с прописной буквы и в конце ставится точка. Пункты списка не должны содержать несколько предложений.

Иллюстрации, за исключением иллюстраций приложений, следует нумеровать

арабскими цифрами без символа «№». Если рисунок один, то он обозначается словом «Рисунок». Например, «Рисунок 1 – Окно сохранения документа». Слово «Рисунок», его номер и наименование располагают по центру строки под изображением без точки в конце.

Нумерация рисунков может быть как сквозная, так и в пределах главы. В последнем случае номер иллюстрации состоит из номера главы и порядкового номера иллюстрации, разделенных точкой, например, «Рисунок 1.1».

Иллюстрации (чертежи, графики, схемы, компьютерные распечатки, диаграммы, фотоснимки) следует располагать в ВКР непосредственно после текста, в котором они упоминаются впервые, или на следующей странице.

Ссылка на рисунок должна предшествовать рисунку. При ссылках на иллюстрации следует писать «… в соответствии с рис. 2» при сквозной нумерации и «…в соответствии с рис. 1.2» при нумерации в пределах главы. Ссылка на рисунок может заключаться в круглые скобки: (рис. 2) или (рис. 1.2), соответственно. Схемы, диаграммы и графики тоже называются рисунками.

Разрешается выполнять иллюстрации в любых цветах на цветном принтере, обеспечивающем хорошее качество печати. Кроме формата А4 для иллюстраций (включая таблицы) разрешается использовать бумагу большего формата вплоть до А3. Такой лист складывается соответствующим образом до формата А4 и рассматривается как приложение. При нумерации он учитывается как одна страница.

Название таблицы следует помещать над таблицей с выравниванием по левому краю с абзацным отступом в одну строку с ее номером и названием через тире, например, «Таблица 2 – Сравнение показателей». Если название таблицы не помещается в одну строку, то на второй строке название начинается под заглавной буквой первой строки. На все таблицы должны быть ссылки в тексте ВКР. При ссылке следует писать слово «таблица» с указанием ее номера, например, «…показано в таблице 2» или (таблица 1.2).

Таблицу с большим количеством строк допускается переносить на другую страницу. При переносе части таблицы на другую страницу слово «Таблица» и ее номер указывают один раз над первой частью таблицы, над другими частями с абзацного отступа пишут «Продолжение таблицы» и указывают ее номер, например, «Продолжение таблицы 1». При переносе таблицы на другую страницу заголовок помещают только над ее первой частью.

Таблицы, за исключением таблиц приложений, следует нумеровать арабскими цифрами сквозной нумерацией. Допускается нумерация таблиц в пределах главы. В этом случае номер таблицы состоит из номера главы и порядкового номера таблицы, разделенных точкой, например, таблица 1.2.

Таблицы каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой буквенного обозначения приложения, например, «Таблица В.1».

Заголовки граф (столбцов) и строк таблицы следует писать с прописной буквы в единственном числе, а подзаголовки граф – со строчной буквы, если они составляют одно предложение с заголовком, или с прописной буквы, если они имеют самостоятельное значение. В конце заголовков и подзаголовков таблиц точки не ставятся. Заголовки граф, как правило, записываются параллельно строкам таблицы. При необходимости допускается перпендикулярное расположение заголовков граф.

Таблицы располагают только после текста, в котором идет о них речь, или на следующей странице. Ни в рисунках, ни в таблицах не должно быть элементов, о которых не идет речь в дипломной работе.

Формулы выводятся на свободные строки (по центру) и могут сопровождаться экспликацией, отделяемой от формулы запятой. В экспликации разъясняется смысл величин и коэффициентов в той последовательности, как они стоят в формуле. Первая строка экспликации начинается словом «где» (двоеточие после него не ставится), затем обозначение первой величины и через тире его расшифровка. После каждой расшифровки ставится точка с запятой, а после последней – точка.

Пример:

 , (2.6)

где (x_0, y_0, z_0) - координаты точки на плоскости.

Номер формулы, заключенный в круглые скобки, ставится на последней строке формулы и обозначается двумя числами – номером главы и порядковым номером в данной главе. Например, «(2.5)» - вторая глава, пятая формула. Если же в ВКР принята сквозная нумерация формул, то в круглых скобках отмечается очередной номер, например, (32). Формула сопровождается номером только в случае, если на нее имеется ссылка в последующем тексте.

Если формула не умещается в одну строку, то она должна быть перенесена после математического знака на другую строку с повторением этого знака в следующей строке.

Формулы, помещаемые в приложениях, должны нумероваться отдельной нумерацией арабскими цифрами в пределах каждого приложения с добавлением перед каждой цифрой обозначения приложения, например, формула «(В.1)». Ссылки в тексте на порядковые номера формул дают в скобках, например, «… в формуле (1)».

Математические знаки "+", "–", ">", "<" используются только в формулах, таблицах и рисунках. В тексте данные знаки должны быть обозначены словами "плюс", "минус", "больше", "меньше" и так далее. Если в тексте приводится диапазон изменений какой-либо величины, то обозначение единиц указывается только после последнего диапазона, например, "... отклонения величин лежат в диапазоне от 8 до 12%... ", или "... отклонения величин лежат в диапазоне 8–12%...". Не допускается отделять единицу величины от числового значения (переносить ее на другую строку или другую страницу).

Между последней цифрой числа и обозначением единицы следует оставлять пробел. Исключения составляют обозначения в виде знака, поднятого над строкой, например 80%, 20˚ и так далее. Единица величины одного и того же параметра в пределах всей работы должна быть постоянной.

Оформление ссылок на литературные источники

Цитаты (выдержки) из источников и литературы используются в тех случаях, когда свою мысль хотят подтвердить точной выдержкой по определенному вопросу. Цитаты должны быть текстуально точными и заключены в кавычки. Если в цитату берется часть текста, то есть не с начала фразы или с пропусками внутри цитируемой части, то место пропуска обозначается отточиями (тремя точками). В тексте необходимо указать в квадратных скобках порядковый номер источника в списке литературы и номер процитированной страницы. Например: [5, 236]. Так делается в случае дословного цитирования. Если же просто ссылаются на соответствующее место в источнике, то перед его номером ставится «См.:». Например: [См.: 11, 118]. При ссылке на источник в конце предложения перед точкой ставится его порядковый номер в квадратных скобках [5], если ссылка содержит несколько источников, то они указываются через запятую [1, 98].

Защита курсовой работы

Защита курсовой работы осуществляется студентом публично перед комиссией из числа представителей профессорско-преподавательского состава кафедры в рамках установленного кафедрой графика. Важными условиями допуска курсовой работы к защите является предоставление чистового текста курсовой работы научному руководителю за две недели до установленной даты защиты, а также наличие отзыва научного руководителя. В котором дается объективная характеристика курсовой работы, отмечаются ее достоинства и недостатки, рекомендованная оценка к защите.. Научный руководитель должен присутствовать на защите курсовой работы.

При выставлении итоговой оценки за курсовую работу для комиссии основополагающими являются указанные выше критерии оценки, а также уровень знаний студента по исследуемой теме и по базовой дисциплине в целом, наличие творческого подхода и самостоятельности в исследовании. Члены комиссии, учитывая мнение научного руководителя, а также общее соответствие работы и защиты критериям, выставляют

итоговую оценку за курсовую работу, научный руководитель в тот же день выставляет ее в ведомость и зачетную книжку студента.

10. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

1. Microsoft Imagine Premium: Microsoft Windows Professional 7
2. Microsoft Office 2007 Russian Academic OPEN No Level
3. Антивирусное программное обеспечение Kaspersky Security
4. ОС Linux
5. GNU gcc
6. LibreOffice

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Вид занятия	***Наименование аудитории***	***Перечень основного оборудования***	***№ ЛР***
1	**2**	**3**	**4**
Лк	Лекционная аудитория	-	-
ЛР	Лаборатория параллельных вычислений	Оборудование 14-ПК i5-2500/H67/4Gb/500Gb (монитор TFT19 Samsung E1920NR); интерактивная доска Smart Board X885ix со встроенным проектором UX60	№ 1-6
	Лаборатория технических средств защиты информации	Оборудование 16-ПК i5-2500/H67/4Gb/500Gb (монитор TFT19 Samsung E1920NR); интерактивная доска Smart Board X885ix со встроенным проектором UX60	№ 7-18
КР	Лаборатория параллельных вычислений	Оборудование 14-ПК i5-2500/H67/4Gb/500Gb (монитор TFT19 Samsung E1920NR)	-
СР	ЧЗ1	Оборудование 10-ПК i5-2500/H67/4Gb (монитор TFT19 Samsung); принтер HP LaserJet P2055D	-

Приложение 1

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ОПК-3	Способность к разработке алгоритмических и программных решений	**1**. Командно-ориентированное программирование	1.1. Обзор языков програмиирования	Индивидуальное задание, вопрос к зачету
			1.2. Синтаксис и семантика языка	Индивидуальное задание, вопрос к зачету
			1.3. Основные алгоритмические конструкции	Индивидуальное задание, вопрос к зачету
ОПК-4	Способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий Способность решать стандартные задачи профессиональной деятельности		1.4. Массивы и указатели	Индивидуальное задание, вопрос к зачету
			1.5. Функции	Индивидуальное задание, вопрос к зачету
			1.6. Структуры	Индивидуальное задание, вопрос к зачету
ПК-9	Способность составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы Способность составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы	**2.** Объектно-ориентированное программирование	2.1 Концепция объектно-ориентированного программирования	Индивидуальное задание, вопрос к зачету
			2.2 Основные конструкции языка C#	Индивидуальное задание, вопрос к зачету
			2.3. Классы	Индивидуальное задание, вопрос к зачету
			2.4. Полиморфизм	Индивидуальное задание, вопрос к зачету
			2.5. Событийно-ориентированное программирование	Индивидуальное задание, вопрос к зачету
			2.6. Наследование	Индивидуальное задание, вопрос к зачету
			3.1 Типы значений и ссылочнные типы	Индивидуальное задание, экзаменнационный вопрос
			3.2. Универсальные типы	Индивидуальное задание, экзаменнационный вопрос

		3.Динамические структуры данных и алгоритмы их обработки	3.3. Линейные структуры	Индивидуальное задание, экзаменнационный вопрос
			3.4. Деревья	Индивидуальное задание, экзаменнационный вопрос
			3.5. Табличные структуры	Индивидуальное задание, экзаменнационный вопрос
			3.6. Коллекции	Индивидуальное задание, экзаменнационный вопрос

2. Вопросы к зачету, 2 семестр

№	Компетенции		ВОПРОСЫ К ЗАЧЕТУ	№ и наименование раздела
п/п	Код	Определение		
1	2	3	4	5
1.	**ОПК-3**	Способность к разработке алгоритмических и программных решений в области системного и прикладного программирования,	1.Классификация языков программирования по степени близости к машинному коду.	**1.** Командно-ориентированное программирование
			2.Классификация языков программирования по стилю программирования.	**1.** Командно-ориентированное программирование
			3.Классификация языков программирования по способу трансляции.	**1.** Командно-ориентированное программирование
			4.Система типов языка. Стандартные типы данных. Преобразование типов. Совместимость типов.	**1.** Командно-ориентированное программирование
			5.Структура программы. Основные синтаксические элементы языка.	**1.** Командно-ориентированное программирование
			6.Конструкции ветвления.	**1.** Командно-ориентированное программирование
			7.Циклы.	**1.** Командно-ориентированное программирование
2.	ОПК-4	Способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической	8.Массивы. Определение. Объявление и инициализация массива.	**1.** Командно-ориентированное программирование
			9.Объявление и описание функций. Прототип функции.	**1.** Командно-ориентированное программирование
			10. Передача аргументов в функцию. Фактические и формальные параметры.	**1.** Командно-ориентированное программирование

		культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	11. Объявление строки. Указатель на строку.	**1.** Командно-ориентированное программирование
			12. Стандартные функции для обработки строк.	**1.** Командно-ориентированное программирование
			13. Объявление и инициализация структуры.	**1.** Командно-ориентированное программирование

Вопросы к зачету, 3 семестр

№ п/п	Компетенции		ВОПРОСЫ К ЗАЧЕТУ	№ и наименование раздела
	Код	Определение		
1	2	3	4	5
1.	ОПК-4	Способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	1. Ключевые понятия ООП: наследование, инкапсуляция, полиморфизм.	**2.** Объектно-ориентированное программирование
			2. Пространства имен. Области видимости.	**2.** Объектно-ориентированное программирование
			3. Типы данных в C#. Динамические переменные. Создание и уничтожение динамических переменных.	**2.** Объектно-ориентированное программирование
			4. Циклы и ветвления в C#. Массивы в C#.	**2.** Объектно-ориентированное программирование
2.	ПК-9	Способность составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы	5. Классы и объекты. Создание и уничтожение объекта класса.	**2.** Объектно-ориентированное программирование
			6. Поля класса. Спецификаторы доступа.	**2.** Объектно-ориентированное программирование
			7. Статические члены и статические данные.	**2.** Объектно-ориентированное программирование
			8. Перегрузка операторов .	**2.** Объектно-ориентированное программирование
			9. Делегаты : сигнатура, объявление и создание объекта.	**2.** Объектно-ориентированное программирование
			10. Отправка и перехват событий.	**2.** Объектно-ориентированное программирование
			11. Основные принципы построения иерархии классов в C#. Базовый и производный классы.	**2.** Объектно-ориентированное программирование
			12. Интерфейсы.	**2.** Объектно-ориентированное программирование

Экзаменационные вопросы, 4 семестр

№	Компетенции		ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ	№ и наименование раздела
п/п	Код	Определение		
1	2	3	4	5
1.	ПК-9	Способность составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы	1. Ссылки. Ссылочные типы. Типы значений.	**3.** Динамические структуры данных и алгоритмы их обработки
			2. Использование ref- и out-параметров	**3.** Динамические структуры данных и алгоритмы их обработки
			3. Универсальные классы и методы.	**3.** Динамические структуры данных и алгоритмы их обработки
			4. Виды динамических структур данных.	**3.** Динамические структуры данных и алгоритмы их обработки
			5. Стек. Организация стека.	**3.** Динамические структуры данных и алгоритмы их обработки
			6. Списки. Однонаправленный список. Двунаправленный список.	**3.** Динамические структуры данных и алгоритмы их обработки
			7. Очереди.	**3.** Динамические структуры данных и алгоритмы их обработки
			8. Деревья: основные определения.	**3.** Динамические структуры данных и алгоритмы их обработки
			9. Реализация бинарного дерева.	**3.** Динамические структуры данных и алгоритмы их обработки
			10. Обход дерева.	**3.** Динамические структуры данных и алгоритмы их обработки
			11. Хеширование.	**3.** Динамические структуры данных и алгоритмы их обработки

1. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
знать: ОПК-3 - основные подходы к разработке программ и базовые алгоритмы обработки данных; ОПК-4 - основные информационно-коммуникационные технологи и требования информационной безопасности ПК-9 - основы рационального планирования времени и ресурсов, необходимых для работы **уметь:** ОПК-3 - разрабатывать программные решения в области прикладного программного обеспечения; ОПК-4 - применять информационно-коммуникационные технологии в профессиональной деятельности ПК-9 - разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения **владеть:** ОПК-3 – приемами построения алгоритмических и программных решений ОПК-4 - культурой применения информационно-коммуникационных технологий ПК-9 - инструментами планирования, контроля и оценки результатов	**отлично**	Демонстрирует все показатели компетенций на высоком уровне, а именно: -знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения ; -владеет приемами построения алгоритмических и программных решений; -владеет культурой применения информационно-коммуникационных технологий; -владеет инструментами планирования, контроля и оценки результатов собственной деятельности ;.
	хорошо	Демонстрирует не менее 7 из 9 заявленных показателей компетенций, например: -знает основные подходы к разработке программ и базовые алгоритмы обработки данных; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения ; -владеет приемами построения алгоритмических и программных решений; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности;

собственной деятельности		-знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения ; -владеет культурой применения информационно-коммуникационных технологий; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения ; -владеет инструментами планирования, контроля и оценки результатов собственной деятельности ; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; -владеет приемами построения алгоритмических и программных решений; -владеет культурой применения информационно-коммуникационных технологий.
	удовлетворительно	Демонстрирует освоение не менее 5 из заявленных 9 параметров компетенций, а именно: знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования

		информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения ; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -владеет приемами построения алгоритмических и программных решений; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -владеет культурой применения информационно-коммуникационных технологий; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности;

		-знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -владеет инструментами планирования, контроля и оценки результатов собственной деятельности .
	неудовлетворительно	Демонстрирует менее половины сформированных параметров компетенций.
	зачтено	Демонстрирует освоение не менее 5 из заявленных 9 параметров компетенций, а именно: знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения .
	не зачтено	Демонстрирует освоение менее чем 5 параметров компетенций.

4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и опыта деятельности

Дисциплина Языки и методы программирования направлена на ознакомление обучающихся с основными концепциями языков программирования, принципами разработки программ в свете различных парадигм программирования; на получение теоретических знаний и практических навыков разработки программ для идентификации, формулирования и решения проблем из различных областей науки и производства, а также осуществления поиска, хранения, обработки и анализа информации из различных источников и представления ее в соответствующем виде и для их дальнейшего использования в

практической деятельности.

Изучение дисциплины Языки и методы программирования предусматривает:

- лекции,
- лабораторные работы;
- курсовую работу;
- зачет;
- экзамен;
- самостоятельную работу.

Для фиксирования успешности обучения предусматривается зачет (2,3 семестры) и экзамен (4 семестр).

В ходе освоения раздела 1 «Командно-ориентированное программирование» обучающиеся должны уяснить идеи разработки программ в командном стиле, основные конструкции языка программирования, способы представления данных и знаний.

При изучении раздела 2 «Объектно-ориентированное программирование» обучающиеся должны усвоить принципы объектно-ориентированного подхода к построению программ, основные концепции: инкапсуляцию, наследование, полиморфизм, абстрагирование. Важно научиться построению классов и иерархии классов в приложении, их применение в сложных проектах.

Итогом изучения раздела 3 «Динамические структуры данных и алгоритмы их обработки» является усвоение идей построения и использования различных структур данных, их практической значимости, приобретение навыков построения алгоритмов обработки динамических структур данных

Обучающимся необходимо овладеть навыками и умениями применения изученных методов для разработки и реализации профессионально ориентированных проектов в последующей учебной деятельности.

Овладение ключевыми понятиями является основой усвоения учебного материала по дисциплине.

При подготовке к зачетам и экзамену особое внимание необходимо уделить рекомендациям и замечаниям преподавателей, ведущих аудиторные занятия по дисциплине

В процессе проведения лабораторных занятий происходит закрепление знаний, формирование умений и навыков применения различных методов решения стандартных ситуаций.

Самостоятельную работу необходимо начинать с чтения лекций и учебников.

В процессе консультации с преподавателем обучающийся выясняет наличие пробелов в знаниях и способах решения разных ситуаций.

Работа с литературой является важнейшим элементом в получении знаний по дисциплине. Прежде всего, необходимо воспользоваться списком рекомендуемой по данной дисциплине литературой. Дополнительные сведения по изучаемым темам можно найти в периодической печати и Интернете.

Предусмотрено проведение аудиторных занятий в виде разнообразных тренингов и ситуаций общения в сочетании с внеаудиторной работой.

Приложение 2

АННОТАЦИЯ
рабочей программы дисциплины
Языки и методы программирования

1. Цель и задачи дисциплины

Целью изучения дисциплины является: ознакомление обучающихся с основами теории программирования, развитие навыков работы в различных системах программирования, освоение различных методов, приемов и способов решения задач из различных предметных областей.

Задачами изучения дисциплины являются:

- раскрыть значение дисциплины в общем и профессиональном образовании человека;
- приобретение навыков работы со стандартными и нестандартными алгоритмами решения задач на компьютере;
- сформировать умения и навыки самостоятельного проектирования программ и решения различного рода задач путем применения средств программирования совместно с другими видами программного обеспечения.

2. Структура дисциплины

2.1 Распределение трудоемкости по отдельным видам учебных занятий, включая самостоятельную работу: Лк.-53 час., ЛР-159 час.; СР-184 час.

Общая трудоемкость дисциплины составляет 432 часов, 12 зачетных единиц.

2.2 Основные разделы дисциплины:

1 –Командно-ориентированное программирование.
2 – Объектно-ориентированное программирование.
3 – Динамические структуры данных и алгоритмы их обработки.

3. Планируемые результаты обучения (перечень компетенций)

Процесс изучения дисциплины направлен на формирование следующих компетенций:

ОПК-3 – способность к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, созданию информационных ресурсов глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям;

ОПК-4 – способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности;

ПК -9 – способность составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы.

4. Вид промежуточной аттестации: зачет, экзамен.

Приложение 3

Протокол о дополнениях и изменениях в рабочей программе
на 20___-20___ учебный год

1. В рабочую программу по дисциплине вносятся следующие дополнения:

_____

_____

2. В рабочую программу по дисциплине вносятся следующие изменения:

_____

_____

_____

Протокол заседания кафедры №____ от «___» _____ 20 ____ г.,
(разработчик)

Заведующий кафедрой _____ _____
(подпись) *(Ф.И.О.)*

Приложение 4

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ОПК-3	Способность к разработке алгоритмических и программных решений в области системного и прикладного программирования,	**1**. Командно-ориентированное программирование	1.2. Синтаксис и семантика языка	ЛР№ 1
			1.3. Основные алгоритмические конструкции	ЛР №2
ОПК-4	Способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий		1.4. Массивы и указатели	ЛР№ 3,4
			1.5. Функции	ЛР№5
			1.6. Структуры	ЛР № 6
		2. Объектно-ориентированное программирование	2.2 Основные конструкции языка C#	ЛР№ 7, КР
			2.3. Классы	ЛР№ 8 КР
			2.4. Полиморфизм	ЛР№ 9 , КР
			2.5. Перехват и обработка событий	ЛР№10
ПК-9	Способность составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы		2.6. Наследование	ЛР№11
			2.7 Исключения	ЛР№12
		3.Динамические структуры данных и алгоритмы их обработки	3.1 Типы значений и ссылочные типы	ЛР№13
			3.2. Универсальные типы	ЛР№14
			3.3. Линейные структуры	ЛР№15
			3.4. Деревья	ЛР№16 ,17
			3.5. Табличные структуры	ЛР№18

2. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
знать: ОПК-3 - основные подходы к разработке программ и базовые алгоритмы обработки данных; ОПК-4 - основные информационно-коммуникационные технологи и требования информационной безопасности ПК-9 - основы рационального планирования времени и ресурсов, необходимых для работы **уметь:** ОПК-3 - разрабатывать программные решения в области прикладного программного обеспечения; ОПК-4 - применять информационно-коммуникационные технологии в профессиональной деятельности ПК-9 - разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения **владеть:** ОПК-3 – приемами построения алгоритмических и программных решений ОПК-4 - культурой применения информационно-коммуникационных технологий ПК-9 - инструментами планирования, контроля и оценки результатов	**отлично**	Демонстрирует все показатели компетенций на высоком уровне, а именно: -знает основные подходы к разработке программ и базовые алгоритмы обработки данных; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения ; -владеет приемами построения алгоритмических и программных решений; -владеет культурой применения информационно-коммуникационных технологий; -владеет инструментами планирования, контроля и оценки результатов собственной деятельности ;.
	хорошо	Демонстрирует не менее 7 из 9 заявленных показателей компетенций, а именно: -знает основные подходы к разработке программ и базовые алгоритмы обработки данных; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения ; -владеет приемами построения алгоритмических и программных решений; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных; -знает основные информационно-коммуникационные технологи и требования информационной безопасности;

собственной деятельности		-знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения ; -владеет культурой применения информационно-коммуникационных технологий; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения ; -владеет инструментами планирования, контроля и оценки результатов собственной деятельности .
	удовлетворительно	Демонстрирует освоение не менее 5 из заявленных 9 параметров компетенций, а именно: знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет применять информационно-

		коммуникационные технологии в профессиональной деятельности ; -владеет приемами построения алгоритмических и программных решений.
	неудовлетворительно	Демонстрирует менее половины сформированных параметров компетенций
	зачтено	Демонстрирует освоение не менее 5 из заявленных 9 параметров компетенций, а именно: знает основные подходы к разработке программ и базовые алгоритмы обработки данных ; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет применять информационно-коммуникационные технологии в профессиональной деятельности ; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -умеет разрабатывать, оценивать и реализовывать процессы жизненного цикла программного обеспечения ; или -знает основные подходы к разработке программ и базовые алгоритмы обработки данных; -знает основные информационно-коммуникационные технологи и требования информационной безопасности; -знает основы рационального планирования времени и ресурсов, необходимых для работы; -умеет разрабатывать программные решения в области прикладного программного обеспечения; -владеет приемами построения алгоритмических и программных решений.
	не зачтено	Демонстрирует освоение менее чем 5 параметров компетенций

Программа составлена в соответствии с федеральным государственным образовательным стандартом высшего образования по направлению подготовки 01.03.02 Прикладная математика и информатика от «12» марта 2015 г. № 228

для набора 2015 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «13 » июля 2015 г. № 475

для набора 2016 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «06» июня 2016г. № 429

Программу составили:

Ратинская Е.В., ст. препод. каф. МиФ _____

Рабочая программа рассмотрена и утверждена на заседании кафедры МиФ

от «21» ноября 2018 г., протокол № 3

И.о. зав.выпускающей кафедрой _____ О.И.Медведева

СОГЛАСОВАНО:

И.о. зав.выпускающей кафедрой _____ О.И Медведева.

Директор библиотеки _____ Т.Ф.Сотник

Рабочая программа одобрена методической комиссией ЕН факультета

от «20 » декабря 2018 г., протокол № 4

Председатель методической комиссии факультета _____ М.А. Варданян

СОГЛАСОВАНО:

Начальник
учебно-методического управления _____ Г.П. Нежевец

Регистрационный №_____