МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Российский государственный гуманитарный университет»
(ФГБОУ ВО «РГГУ»)

ИНСТИТУТ ИНФОРМАЦИОННЫХ НАУК И ТЕХНОЛОГИЙ БЕЗОПАСНОСТИ
Факультет информационных систем и безопасности
Кафедра фундаментальной и прикладной математики

СОВРЕМЕННЫЕ СИСТЕМЫ ПРОГРАММИРОВАНИЯ

01.04.04 Прикладная математика

Код и наименование направления подготовки/специальности

Математические методы и модели обработки
и защиты информации в социотехнических системах

Наименование направленности (профиля)/ специализации

Уровень высшего образования: *магистратура*

Форма обучения: *очная, очно-заочная, заочная*

РПД адаптирована для лиц
с ограниченными возможностями
здоровья и инвалидов

Москва 2023

СОВРЕМЕННЫЕ СИСТЕМЫ ПРОГРАММИРОВАНИЯ
Рабочая программа дисциплины

Составитель:
к.т.н., доцент Д.Ю. Клехо

УТВЕРЖДЕНО
Протокол заседания кафедры
фундаментальной и прикладной математики
№ 8 от 06.04.2023

ОГЛАВЛЕНИЕ

1. Пояснительная записка

1.1. Цель и задачи дисциплины

Цель дисциплины - получение базовых знаний о современных методах разработки прикладных программ и информационных систем на основе концепций объектно-ориентированного подхода, а также развитие навыков самостоятельной работы, связанных с анализом, детализацией, выбором методов решения поставленных задач, планированием использования возможностей современных сред программирования, а также различных источников информации для реализации программных приложений.

Задачи:

- получить представление об основных принципах процедурного программирования;
- получить представление об основных понятиях объектно-ориентированного проектирования и программирования;
- получить практические навыки по разработке программного обеспечения.

1.2. Перечень планируемых результатов обучения по дисциплине, соотнесенных с индикаторами достижения компетенций

Компетенция (код и наименование)	**Индикаторы компетенций** (код и наименование)	**Результаты обучения**
ПК-1. Способен проводить систематизацию, алгоритмизацию конкретных информационных потоков по месту научных исследований, производственной деятельности	ПК- 1.1. Переформулирует задачи, данные на естественных языках конкретного научного знания на необходимый язык математики; формулирует теоремы.	***Знать:*** способы описания алгоритма решения вычислительных задач, управляющие конструкции современных языков и правила оформления программного кода; ***Уметь:*** работать с современными системами программирования, применять приемы и методы современного программирования; ***Владеть:*** языками процедурного и объектно-ориентированного программирования, навыками разработки и отладки программ на языках высокого уровня.
	ПК – 1.2. Выделяет динамические, статистические структуры для представления их математическими моделями.	***Знать:*** структуру программы и базовые типы данных; ***Уметь:*** анализировать программу на предмет эффективности человеко-машинного взаимодействия и оптимальности программного решения; ***Владеть:*** навыками оптимизации программного кода.

1.3. Место дисциплины в структуре образовательной программы

Дисциплина «Современные системы программирования» относится к части, формируемой участниками образовательных отношений, блока дисциплин учебного плана.

Дисциплина «Современные системы программирования» имеет своей целью дать базовые знания о современных методах разработки прикладных программ и информационных систем на основе концепций объектно-ориентированного подхода.

В процессе изучения дисциплины студенты приобретают навыки самостоятельной работы, связанные с анализом, детализацией, выбором методов решения поставленных задач,

планированием использования возможностей современных сред программирования, а также различных источников информации для реализации программных приложений.

Для освоения дисциплины необходимы знания, умения и владения, сформированные в ходе изучения следующих дисциплин: «Основы современных технологий коммуникации в социотехнических системах», «Математические методы исследования социальных систем».

В результате освоения дисциплины «Современные системы программирования» формируются знания, умения и владения, необходимые для изучения следующих дисциплин: «Интеллектуальные системы».

2. Структура дисциплины

Общая трудоёмкость дисциплины составляет 4 з.е., 144 академических часа(ов).

Структура дисциплины для очной формы обучения

Объем дисциплины в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении учебных занятий:

Семестр	Тип учебных занятий	Количество часов
3	Лекции	**12**
3	Практические занятия	**28**
	Всего:	**40**

Объем дисциплины (модуля) в форме самостоятельной работы обучающихся составляет 104 академических часа(ов).

Структура дисциплины для очно-заочной формы обучения

Объем дисциплины в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении учебных занятий:

Семестр	Тип учебных занятий	Количество часов
3	Лекции	**10**
3	Практические занятия	**22**
	Всего:	**32**

Объем дисциплины (модуля) в форме самостоятельной работы обучающихся составляет 112 академических часа(ов).

Структура дисциплины для заочной формы обучения

Объем дисциплины в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении учебных занятий:

Семестр	Тип учебных занятий	Количество часов
3	Лекции	**4**
3	Практические занятия	**12**
	Всего:	**16**

Объем дисциплины (модуля) в форме самостоятельной работы обучающихся составляет 128 академических часа(ов).

3. Содержание дисциплины

№	Наименование раздела дисциплины	Содержание
1.	Тема 1. Системы программирования.	Язык и система программирования – понятие, сущность. История развития. Компоненты системы программирования. Современные трансляторы языков программирования. Интерпретация и компиляция. Системы управления памятью и компоновка исполняемых модулей. Библиотеки прикладных программ. Статистические и динамические модули. Текстовые редакторы для создания кода программы. Подсистемы визуальной разработки интерфейса.
2.	Тема 2. Сравнительные характеристики языков программирования	Модель вычислений фон Неймана и традиционные языки программирования. Структура вычислений и структура текста программы. Многопроцессорные вычисления. Статистические и динамические переменные. Модули языка. Организация памяти под структуру классов. Семантика языков программирования. Стили программирования.
3.	Тема 3. Методы программирования	Методы программирования от состояний. Методы, основанные на рекурсии. Объектно-ориентированные методы. Сентенциальные методы. Функциональное программирование. Моделирование.

4. Образовательные технологии

№ п/п	Наименование раздела	Виды учебных занятий	Образовательные технологии
1	2	3	4
1.	Тема 1. Системы программирования.	Лекции 1-2. Практическое занятие. Самостоятельная работа.	Лекция с использованием видеоматериалов Выполнение практической работы при помощи специализированного ПО. Консультирование и проверка самостоятельной работы посредством электронной почты.
2.	Тема 2. Сравнительные характеристики языков программирования.	Лекции 3-4 Практическое занятие.	Лекция с использованием видеоматериалов, опрос. Выполнение практической работы при помощи специализированного ПО.

		Самостоятельная работа.	Консультирование и проверка самостоятельной работы посредством электронной почты.
3.	Тема 3. Методы программирования.	Лекции 5-6 Практическое занятие. Самостоятельная работа.	Лекция с использованием видеоматериалов. Выполнение практической работы при помощи специализированного ПО. Консультирование и проверка самостоятельной работы посредством электронной почты.

В период временного приостановления посещения обучающимися помещений и территории РГГУ для организации учебного процесса с применением электронного обучения и дистанционных образовательных технологий могут быть использованы следующие образовательные технологии:

– видео-лекции**;**

– онлайн-лекции в режиме реального времени;

– электронные учебники, учебные пособия, научные издания в электронном виде и доступ к иным электронным образовательным ресурсам;

– системы для электронного тестирования;

– консультации с использованием телекоммуникационных средств.

5. Оценка планируемых результатов обучения

5.1 Система оценивания

Форма контроля	**Макс. количество баллов**	
	За одну работу	**Всего**
Текущий контроль: - опрос - защита практической работы 1-3 - самостоятельная работа	 *10 баллов* *10 баллов* *10 баллов*	 *20 баллов* *30 баллов* *10 баллов*
Промежуточная аттестация - зачет (ответы на вопросы)		*40 баллов*
Итого за семестр		*100 баллов*

Полученный совокупный результат конвертируется в традиционную шкалу оценок и в шкалу оценок Европейской системы переноса и накопления кредитов (European Credit Transfer System; далее – ECTS) в соответствии с таблицей:

100-балльная шкала	Традиционная шкала		Шкала ECTS
95 – 100	отлично	зачтено	A
83 – 94			B
68 – 82	хорошо		C
56 – 67	удовлетворительно		D
50 – 55			E
20 – 49	неудовлетворительно	не зачтено	FX
0 – 19			F

5.2 Критерии выставления оценки по дисциплине

Баллы/ Шкала ECTS	Оценка по дисциплине	Критерии оценки результатов обучения по дисциплине
100-83/ A,B	отлично/ зачтено	Выставляется обучающемуся, если он глубоко и прочно усвоил теоретический и практический материал, может продемонстрировать это на занятиях и в ходе промежуточной аттестации. Обучающийся исчерпывающе и логически стройно излагает учебный материал, умеет увязывать теорию с практикой, справляется с решением задач профессиональной направленности высокого уровня сложности, правильно обосновывает принятые решения. Свободно ориентируется в учебной и профессиональной литературе. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне – «высокий».
82-68/ C	хорошо/ зачтено	Выставляется обучающемуся, если он знает теоретический и практический материал, грамотно и по существу излагает его на занятиях и в ходе промежуточной аттестации, не допуская существенных неточностей. Обучающийся правильно применяет теоретические положения при решении практических задач профессиональной направленности разного уровня сложности, владеет необходимыми для этого навыками и приёмами. Достаточно хорошо ориентируется в учебной и профессиональной литературе. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне – «хороший».
67-50/ D,E	удовлетво-рительно/ зачтено	Выставляется обучающемуся, если он знает на базовом уровне теоретический и практический материал, допускает отдельные ошибки при его изложении на занятиях и в ходе промежуточной аттестации. Обучающийся испытывает определённые затруднения в применении теоретических положений при решении практических задач профессиональной направленности стандартного уровня сложности, владеет необходимыми для этого базовыми навыками и приёмами. Демонстрирует достаточный уровень знания учебной литературы по дисциплине. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне – «достаточный».
49-0/ F,FX	неудовлет-ворительно/ не зачтено	Выставляется обучающемуся, если он не знает на базовом уровне теоретический и практический материал, допускает грубые ошибки при его изложении на занятиях и в ходе промежуточной аттестации. Обучающийся испытывает серьёзные затруднения в применении теоретических положений при решении практических задач профессиональной направленности стандартного уровня сложности, не владеет необходимыми для этого навыками и приёмами. Демонстрирует фрагментарные знания учебной литературы по дисциплине. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции на уровне «достаточный», закреплённые за дисциплиной, не сформированы.

5.3 Оценочные средства (материалы) для текущего контроля успеваемости, промежуточной аттестации обучающихся по дисциплине

Вопросы на опрос по лекции:

1) Задачи человеко-компьютерного взаимодействия.
2) Понятие ЧМИ.

3) Эргономические принципы ЧМИ.
4) Законы восприятия информации человеком.
5) Цветовые модели. Их классификация и различия.
6) Рациональное размещение данных на экране.
7) Основные понятия и стандарты человеко-машинных интерфейсов.
8) Базовые элементы управления диалогового окна.
9) Элементы управления приложения.
10) Меню.
11) Свойства интерфейса.
12) Показатели качества.
13) Концептуальное проектирование Web-приложений.
14) Проектирование схемы навигации.
15) Проектирование дизайна.
16) Разработка макета.
17) Разработка стилей проекта.

Вопросы для самостоятельной работы.

1. Сравнительный анализ машинно-ориентированных языков программирования.
2. Структура вычислений и структура текста программы. Многопроцессорные вычисления.
3. Статистические и динамические переменные. Модули языка.
4. Организация памяти под структуру классов.
5. Семантика языков программирования.
6. Стили программирования.
7. Методы программирования.
8. Структуры программирования.
9. Структуры данных.
10. Объектно-ориентированный подход.
11. Разработка учебного проекта.
12. Отладка учебного проекта.

Примерные вопросы на промежуточную аттестацию (зачет):

ПК – 1.1. - *переформулирует задачи, данные на естественных языках конкретного научного знания на необходимый язык математики; формулирует теоремы.*

1. Органы управления. Их классификация
2. Типы ЧМИ.
3. Принципы построения ЧМИ
4. Специализированные устройства управления
5. Проектирование интерфейса в модели ЖЦ ПО.
6. Прототипирование.
7. Управление устройствами.
8. Клавиатура.
9. Мышь.
10. Управление монитором.
11. Интерфейсы систем управления технологическими процессами.
12. Мультимедиа среды.
13. Мультисенсорные системы.
14. Бесконтактные жестовые интерфейсы.
15. Естественно-языковое взаимодействие.

ПК – 1.2. - *Выделяет динамические, статистические структуры для представления их математическими моделями.*

16. Считывание телодвижений (косвенный ввод данных).
17. Особенности технологии MVC.
18. Язык Razor.
19. Трехуровневая схема проектирования.
20. Уровни автоматизации технологических процессов и уровни контроля пользователя.

6. Учебно-методическое и информационное обеспечение дисциплины

6.1 Список источников и литературы

Литература

Основная

1. Гуськова, О.И. Объектно ориентированное программирование в Java : учебное пособие / О. И. Гуськова. - Москва : МПГУ, 2018. - 240 с. - ISBN 978-5-4263-0648-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1020593 – Режим доступа: по подписке.

2. Шелудько, В. М. Язык программирования высокого уровня Python. Функции, структуры данных, дополнительные модули : учебное пособие / В. М. Шелудько ; Южный федеральный университет. - Ростов-наДону ; Таганрог : Издательство Южного федерального университета, 2017. - 107 с. - ISBN 978-5-9275-2648-2. - Текст : электронный. - URL: https://znanium.com/catalog/product/1021664.

3. Титов, Д. В. Электронное администрирование в государственном управлении : учебное пособие / Д. В. Титов, А. Н. Наимов ; Федер. служба исполн. наказаний, Вологод. ин-т права и экономики. - Вологда : ВИПЭ ФСИН России, 2019. - 78 с. - ISBN 978-5-94991-493-9. - Текст: электронный. - URL: https://znanium.com/catalog/product/1229826.

Дополнительная

1. Федотов, И. Е. Модели параллельного программирования: Практическое пособие / Федотов И.Е. - Москва :СОЛОН-Пр., 2017. - 392 с. (Библиотека профессионала)ISBN 978-5-91359-222-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/858609.

2. Богданов, Е. П. Интеллектуальный анализ данных : практикум для магистрантов направления 09.04.03 «Прикладная информатика» профиль подготовки «Информационные системы и технологии корпоративного управления» / Е. П. Богданов. - Волгоград : ФГБОУ ВО Волгоградский ГАУ, 2019. - 112 с. - Текст: электронный. - URL: https://znanium.com/catalog/product/1087885.

3. Андрейчиков, А. В. Интеллектуальные цифровые технологии концептуального проектирования инженерных решений : учебник / А. В. Андрейчиков, О. Н. Андрейчикова. — Москва : ИНФРА-М, 2019. — 511 с. — (Высшее образование: Магистратура). - ISBN 978-5-16-014884-7. - Текст: электронный. - URL: https://znanium.com/catalog/product/1009598.

6.2 Перечень ресурсов информационно-телекоммуникационной сети «Интернет».

1. Электронно-библиотечная система - https://new.znanium.com/
2. Портал Министерства образования и науки - http://www.edu.ru/
3. Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации - https://digital.gov.ru/ru /

Национальная электронная библиотека (НЭБ) www.rusneb.ru

ELibrary.ru Научная электронная библиотека www.elibrary.ru

6.3 Профессиональные базы данных и информационно-справочные системы

Доступ к профессиональным базам данных: https://liber.rsuh.ru/ru/bases

Информационные справочные системы:
1. Консультант Плюс
2. Гарант

7. Материально-техническое обеспечение дисциплины

- Для обеспечения дисциплины используется материально-техническая база образовательного учреждения:
- лаборатория или компьютерный класс,
- доска,
- проектор (стационарный или переносной),
- компьютер или ноутбук для преподавателя,
- компьютеры для обучающихся,

Состав программного обеспечения:
1. Windows
2. Microsoft Office
3. Kaspersky Endpoint Security
4. CorelDrawCS6
5. Adobe CS4 Master Collection
6. Microsoft Visual Professional 2019
7. Microsoft SQL Server 2008
8. Mozilla Firefox

8. Обеспечение образовательного процесса для лиц с ограниченными возможностями здоровья и инвалидов

В ходе реализации дисциплины используются следующие дополнительные методы обучения, текущего контроля успеваемости и промежуточной аттестации обучающихся в зависимости от их индивидуальных особенностей:

- для слепых и слабовидящих: лекции оформляются в виде электронного документа, доступного с помощью компьютера со специализированным программным обеспечением; письменные задания выполняются на компьютере со специализированным программным обеспечением или могут быть заменены устным ответом; обеспечивается индивидуальное равномерное освещение не менее 300 люкс; для выполнения задания при необходимости предоставляется увеличивающее устройство; возможно также использование собственных увеличивающих устройств; письменные задания оформляются увеличенным шрифтом; экзамен и зачёт проводятся в устной форме или выполняются в письменной форме на компьютере.
- для глухих и слабослышащих: лекции оформляются в виде электронного документа, либо предоставляется звукоусиливающая аппаратура индивидуального пользования; письменные задания выполняются на компьютере в письменной форме; экзамен и зачёт проводятся в письменной форме на компьютере; возможно проведение в форме тестирования.
- для лиц с нарушениями опорно-двигательного аппарата: лекции оформляются в виде электронного документа, доступного с помощью компьютера со специализированным

программным обеспечением; письменные задания выполняются на компьютере со специализированным программным обеспечением; экзамен и зачёт проводятся в устной форме или выполняются в письменной форме на компьютере.

При необходимости предусматривается увеличение времени для подготовки ответа.

Процедура проведения промежуточной аттестации для обучающихся устанавливается с учётом их индивидуальных психофизических особенностей. Промежуточная аттестация может проводиться в несколько этапов.

При проведении процедуры оценивания результатов обучения предусматривается использование технических средств, необходимых в связи с индивидуальными особенностями обучающихся. Эти средства могут быть предоставлены университетом, или могут использоваться собственные технические средства.

Проведение процедуры оценивания результатов обучения допускается с использованием дистанционных образовательных технологий.

Обеспечивается доступ к информационным и библиографическим ресурсам в сети Интернет для каждого обучающегося в формах, адаптированных к ограничениям их здоровья и восприятия информации:

- для слепых и слабовидящих: в печатной форме увеличенным шрифтом, в форме электронного документа, в форме аудиофайла.
- для глухих и слабослышащих: в печатной форме, в форме электронного документа.
- для обучающихся с нарушениями опорно-двигательного аппарата: в печатной форме, в форме электронного документа, в форме аудиофайла.

Учебные аудитории для всех видов контактной и самостоятельной работы, научная библиотека и иные помещения для обучения оснащены специальным оборудованием и учебными местами с техническими средствами обучения:

- для слепых и слабовидящих: устройством для сканирования и чтения с камерой SARA CE; дисплеем Брайля PAC Mate 20; принтером Брайля EmBraille ViewPlus;
- для глухих и слабослышащих: автоматизированным рабочим местом для людей с нарушением слуха и слабослышащих; акустический усилитель и колонки;
- для обучающихся с нарушениями опорно-двигательного аппарата: передвижными, регулируемыми эргономическими партами СИ-1; компьютерной техникой со специальным программным обеспечением.

9. Методические материалы

9.1 Планы практических занятий

Практическая работа 1. Реализация простейшей программы на VC++, с использованием интерфейса и библиотеки CLR.

Задания.

1. Научиться разрабатывать и реализовывать простейшие программы на языке VC++.
2. Получить практические навыки работы по использованию интерфейса CLR.
3. Научиться связывать переменные и методы с элементами диалогового окна.

Указания по выполнению заданий:

1. Ввод данных через текстовое поле TextBox с проверкой типа методом TryParse При работе с формой очень часто ввод данных организуют через элемент управления текстовое поле TextBox . Напишем типичную программу, которая вводит через текстовое поле число, при нажатии командной кнопки извлекает из него квадратный корень и выводит результат на метку Label. В случае ввода не числа сообщает пользователю об этом.

2. Решая сформулированную задачу, запускаем Visual Studio, выбираем пункт меню File-New -Project. В окне New Project в узле Visual C++ выберем среду CLR, а затем в области шаблоны выберем шаблон (Templates) Windows Forms Application Visual C++. В качестве имени проекта введем имя Корень и щелкнем на кнопке OK.

3. Далее из панели элементов управления Toolbox (если в данный момент вы не видите панель элементов управления, то ее можно добавить, например, с помощью комбинации клавиш Ctrl+Alt+x или меню View - oolbox) в форму с помощью указателя мыши перетаскиваем текстовое поле TextBox, метку Label и командную кнопку Button.

4. Получить названные элементы на проектируемой экранной форме можно, также дважды щелкая указателем мыши на каждом элементе в панели Tools. Таким образом, в форме будут находиться три элемента управления. Расположим их на экранной форме.

5. Теперь следует изменить некоторые свойства элементов управления. Чтобы получить пустой обработчик загрузки формы, дважды щелкнем по проектируемой экранной форме. Сразу после этого мы попадаем на вкладку программного кода Form1.h. Здесь задаем свойствам формы (к форме обращаемся посредством ссылки this), кнопкам button1 и текстового поля textBox1, метке label1 следующие значения:

this->Text = "Извлечение квадратного корня"; button1->Text = "Извлечь корень";

textBox1->Clear(); // Очистка текстового поля label1->Text = nullptr; // или = String::Empty;

6. Нажмем клавишу F5 для выявления возможных опечаток, то есть синтаксических ошибок и предварительного просмотра дизайна будущей программы (рис. 1).

7. Далее программируем событие button1_Click — «щелчок мышью на кнопке Извлечь корень». Создать пустой обработчик этого события удобно, дважды щелкнув мышью на этой кнопке. Между двумя появившимися строчками программируем диагностику правильности вводимых данных, конвертирование строковой переменной в переменную типа Single и непосредственное извлечение корня (листинг).

Листинг. Фрагмент программы извлечения корня с проверкой типа методом TryParse

```
// ..............
//  Программный код, расположенный выше, создан средой
    Visual Studio
// автоматически, поэтому автором не
приводится this->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion
// Программа вводит через текстовое поле число, при щелчке на
командной
//  кнопке извлекает из него квадратный корень и
    выводит результат
//  на метку label1. В случае ввода не числа
    сообщает пользователю об
// этом, выводя красным цветом предупреждение также на
   метку label1.
private: System::
```

```
Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
button1->Text = "Извлечь
корень"; label1->Text =
String::Empty;
// или label1->Text = nullptr;
this->Text = "Извлечение квадратного корня";
textBox1->Clear(); // Очистка текстового
поля
textBox1->TabIndex = 0; // Установка фокуса в текстовом поле
}
private:  System::  //  Обработка  щелчка  на  кнопке
          "Извлечь корень":
Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
Single X; // - из этого числа будем извлекать корень
// Преобразование из строковой переменной в
Single: bool Число_ли = Single::TryParse(textBox1-
>Text,
System::Globalization::NumberStyles::Number,
System::Globalization::NumberFormatInfo::CurrentInfo, X);
// Второй параметр - это разрешенный стиль числа (Integer,
//  шестнадцатеричное  число,  экспоненциальный  вид  числа
    и прочее).
// Третий параметр форматирует значения на основе текущего языка
// и региональных параметров из Панели управления - Язык и
// региональные стандарты - число допустимого формата; метод
// возвращает значение в переменную
X if (Число_ли == false)
{ // Если пользователь ввел не число:
label1->Text = "Следует вводить числа";
label1->ForeColor = Color::Red; // - цвет текста на
метке return; // - выход из процедуры
}
Single Y = (Single)Math::Sqrt(X); // - извлечение корня
label1->ForeColor = Color::Black; // - черный цвет
текста на метке
label1->Text = String::Format("Корень из {0} равен {1:F5}", X,
Y);
}
};
}
```

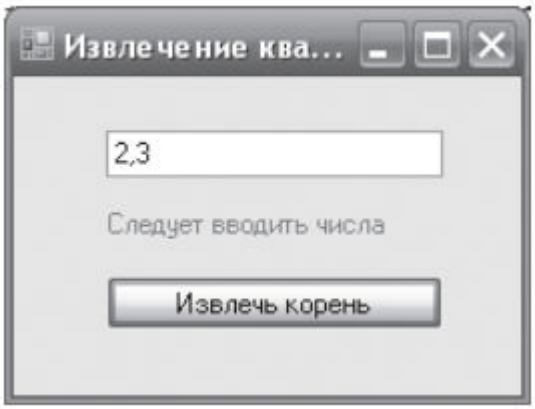

Рис.1 Фрагмент работы программы.

8. Если пользователь ввел все-таки число, то будет выполняться следующий оператор извлечения квадратного корня Math::Sqrt(X). Математические функции Visual Studio являются методами класса Math. Их можно увидеть, набрав Math и введя так называемый *оператор разрешения области действия* (::). В раскрывающемся списке вы увидите множество математических функций: Abs, Sin, Cos, Min и т. д. и два свойства — две константы E = 2,718... (основание натуральных логарифмов) и PI = 3,14... (число диаметров, уложенных вдоль окружности). Функция Math::Sqrt(X)возвращает значение типа double (двойной точности с плавающей запятой), которое мы *приводим* с помощью неявного преобразования (Single) к переменной одинарной точности.

9. Последней строчкой обработки события button1_Click является формирование строки label1->Text с использованием метода String::Format. Использованный формат ≪Корень из {0} равен {1:F5}≫ означает: взять нулевой выводимый элемент, то есть переменную X, и записать эту переменную вместо фигурных скобок; после чего взять первый выводимый элемент, то есть Y, и записать его вместо вторых фигурных скобок в формате с фиксированной точкой и пятью десятичными знаками после запятой.

10. Нажав клавишу F5, проверяем, как работает программа. Результат работающей программы представлен на рис. 1.2.

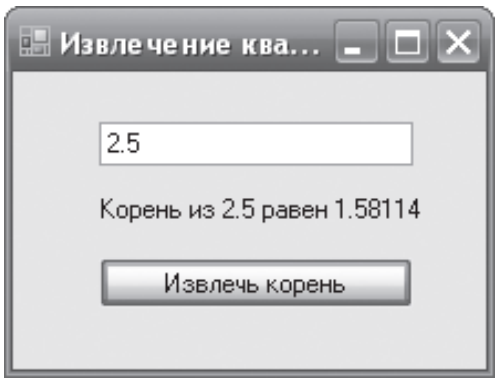

Рис.2.2. Извлечение квадратного корня

Контрольные вопросы.

1. Опишите шаги, которые вы сделали, чтобы открыть диалоговую панель программы для ее визуальной настройки.
2. Что такое Class Wizard.

3. Как добавить элемент для ввода данных.

Практическая работа 2. Создание графического интерфейса на базе диалогового окна VC++.

Задание:

1. Научиться разрабатывать и реализовывать программу, использующую вкладки и переключатели.
2. Изменять шрифт вкладок.

Указания по выполнению заданий:

1. Разработать программу, позволяющую выбрать текст из двух вариантов, задать цвет и размер шрифта этого текста на трех вкладках TabControl с использованием переключателей RadioButton.
2. Программируя поставленную задачу, запустим Visual Studio и выберем приложение в среде CLR шаблона Windows Forms Application Visual C++. Назовем этот проект Вкладки.
3. Используя панель элементов Toolbox, в форму перетащим с помощью мыши элемент управления TabControl. Как видно, по умолчанию имеем две вкладки, а по условию задачи, как показано на рисунке, три вкладки. Добавить третью вкладку можно в конструкторе формы, а можно программно (рис.2.1).

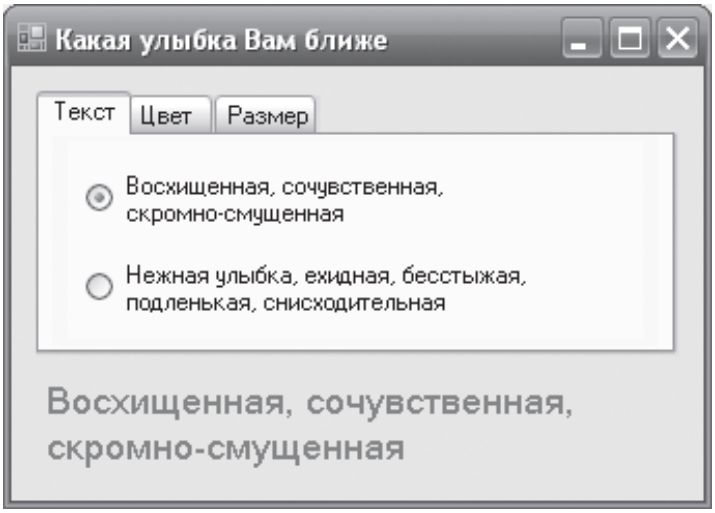

Рис.2.1 Программа с переключателями и вкладками

4. Чтобы добавить третью вкладку в конструкторе, необходимо в свойствах (окно Properties) элемента управления tabControl1 выбрать свойство TabPages, в результате попадаем в диалоговое окно TabPage Collection Edit, где добавляем (кнопка Add) третью вкладку (первые две присутствуют по умолчанию). Эти вкладки нумеруются от нуля, то есть третья вкладка будет распознаваться как TabPages(2). Название каждой вкладки будем указывать в программном коде.
5. Рассмотрим, как добавить третью вкладку не в конструкторе, а в программном коде при обработке события загрузки формы (листинг). Однако прежде чем перейти на вкладку программного кода, для каждой вкладки выбираем из панели Toolbox по два переключателя RadioButton, а в форму перетаскиваем метку Label. Теперь с помощью щелчка правой кнопкой мыши в пределах формы переключаемся на редактирование программного кода.

Листинг Фрагмент программы, управляющей вкладками и переключателями.

```
// ..............
// Программный код, расположенный выше, создан средой
    Visual Studio
// автоматически, поэтому автором не
приводится this->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion
// Программа, позволяющая выбрать текст из двух
    вариантов, задать цвет
// и размер шрифта для этого текста на трех вкладках TabControl
// с использованием переключателей
RadioButton private: System::
Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
// Создание третьей вкладки "программно":
auto tabPage3 = gcnew
System::Windows::Forms::TabPage(); tabPage3-
>UseVisualStyleBackColor = true;
// Добавление третьей вкладки в существующий набор
// вкладок tabControl1:
this->tabControl1->Controls->Add(tabPage3);
// Добавление переключателей 5 и 6 на третью
вкладку: tabPage3->Controls->Add(this-
>radioButton5); tabPage3->Controls->Add(this-
>radioButton6);
// Расположение переключателей 5 и 6:
this->radioButton5->Location = System::Drawing::Point(20,
15); this->radioButton6->Location =
System::Drawing::Point(20, 58); this->Text = "Какая улыбка
вам ближе";
// Задаем названия вкладок:
tabControl1->TabPages[0]->Text = "Текст";
tabControl1->TabPages[1]->Text = "Цвет";
tabControl1->TabPages[2]->Text =
"Размер";
// Эта пара переключателей изменяет
текст: radioButton1->Text =
"Восхищенная, сочувственная,\nскромно-смущенная";
radioButton2->Text = "Нежная улыбка, ехидная, бес"
+ "стыжая,\nподленькая, снисходительная";
// или
// radioButton2->Text = "Нежная улыбка, бесстыжая," +
// Environment::NewLine + "подленькая, снисходительная";
// Эта пара переключателей изменяет цвет
текста: radioButton3->Text = "Красный";
radioButton4->Text = "Синий";
// Эта пара переключателей изменяет размет
шрифта: radioButton5->Text = "11 пунктов";
radioButton6->Text = "13
пунктов"; label1->Text =
radioButton1->Text;
}
```

```
private:                                                System::Void
radioButton1_CheckedChanged(System::Object^
sender, System::EventArgs^ e)
{ label1->Text = radioButton1->Text; }
private:                                                System::Void
radioButton2_CheckedChanged(System::Object^
sender, System::EventArgs^ e)
{ label1->Text = radioButton2->Text; }
private:                                                System::Void
radioButton3_CheckedChanged(System::Object^
sender, System::EventArgs^ e)
{ label1->ForeColor = Color::Red; }
private:                                                System::Void
radioButton4_CheckedChanged(System::Object^
sender, System::EventArgs^ e)
{ label1->ForeColor = Color::Blue; }
private:                                                System::Void
radioButton5_CheckedChanged(System::Object^
sender, System::EventArgs^ e)
{ label1->Font = gcnew System::Drawing::
Font(label1->Font->Name, 11); }
private:                                                System::Void
radioButton6_CheckedChanged(System::Object^
sender,System::EventArgs^ e)
{ label1->Font = gcnew
System::Drawing:: Font(label1->Font-
>Name, 13); }
};
}
```

6. Как видно из текста программы, при обработке события загрузки формы Form1_Load (этот участок программного кода можно было бы задать сразу после вызова процедуры InitializeComponent) создаем «программно» третью вкладку. Заметьте, что мы ее объявили как auto, то есть тип переменной tabPage3 выводится из выражения инициализации в Visual C++. Далее добавляем новую вкладку tabPage3 в набор вкладок tabControl1, созданный в конструкторе. Затем «привязываем» пятый и шестой переключатели к третьей вкладке.

Каждая пара переключателей, расположенных на каком-либо элементе управления (в данном случае на различных вкладках), «отрицают» друг друга, то есть если пользователь выбрал один, то другой переходит в противоположное состояние. Отслеживать изменения состояния переключателей удобно с помощью обработки событий переключателей CheckChanged (см. листинг).

7. Чтобы получить пустой обработчик этого события в конструкторе формы, следует дважды щелкнуть на соответствующем переключателе и таким образом запрограммировать изменения состояния переключателей.

Контрольные вопросы.

1. Опишите, как добавить в программу вкладки.
2. По какому принципу работают переключатели RadioButton.
3. Опишите, как организовать работу группы переключателей.

Практическая работа 3. Программирование консольных приложений.
Задание:

1. Научиться программировать консольное приложение, для вычисления математических функций, вводимых значений и вывода результата на экран.
2. Получить практические навыки по использованию различных элементов графического интерфейса и операторов языка VC++.

Указания по выполнению заданий:

1. Иногда, например для научных расчетов, требуется организовать какой-нибудь самый простой ввод данных, выполнить весьма сложную математическую обработку введенных данных и оперативно вывести на экран результат вычислений.

Можно по-разному организовать такую программу, в том числе программируя так называемое *консольное приложение* (от англ. *console* — пульт управления). Под консолью обычно подразумевают экран компьютера и клавиатуру.

2. Напишем консольное приложение, которое приглашает пользователя ввести два числа, складывает их и выводит результат вычислений на консоль. Для этого запускаем Visual C++ 2010, далее создаем новый проект (New Project), в узле Visual C++ в среде CLR выбираем шаблон Console Application CLR, задаем имя решения (Name) — Сумма. После щелчка на кнопке OK попадаем сразу на вкладку программного кода (рис. 3.1).

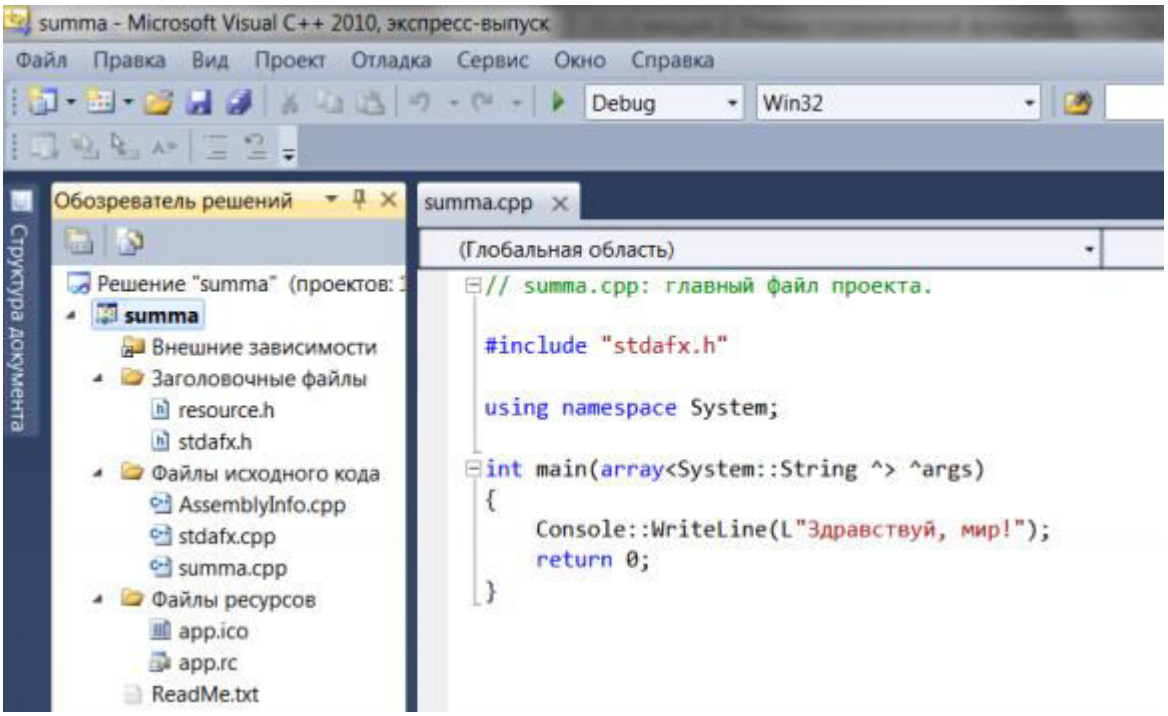

Рис. 3.1. Вкладка программного кода.

3. Как видите, здесь управляющая среда Visual Studio приготовила несколько строк программного кода. Это вполне работоспособная программа. При запуске консольного или Windows-приложения C++ метод Main() является первым вызываемым методом. В фигурные скобки после Main() мы вставим собственный программный код (листинг). Фрагмент работы программы на рисунке 3.2.

Листинг Ввод и вывод данных в консольном приложении

```
    // Сумма.cpp: главный файл проекта.
    // Программа организует ввод двух чисел, их сложение и
вывод суммы на консоль
  #include "stdafx.h" using namespace System;
    int main(array<System::String ^> ^args)
    {
    // Задаем строку заголовка консоли:
    Console::Title = "Складываю два числа:";
Console::BackgroundColor = ConsoleColor::Cyan; // - цвет фона
Console::ForegroundColor = ConsoleColor::Black; // - цвет
текста Console::Clear();
    // Ввод первого слагаемого:
Console::WriteLine("Введите первое
слагаемое:"); String^ Строка =
Console::ReadLine();
    Single X, Y, Z;
    // Преобразование строковой переменной в
число: X = Single::Parse(Строка);
    // Ввод второго слагаемого:
Console::WriteLine("Введите второе
слагаемое:"); Строка = Console::ReadLine();
    Y =
Single::Parse(Строка); Z
= X + Y;
    Console::WriteLine("Сумма = {0} + {1} = {2}", X, Y, Z);
    // Звуковой сигнал частотой 1000 Гц и длительностью 0.5
секунды: Console::Beep(1000, 500);
    //          Приостановить          выполнение   программы
   до           нажатия     какой-нибудь клавиши:
    Console::Read
Key(); return 0;
    }
```

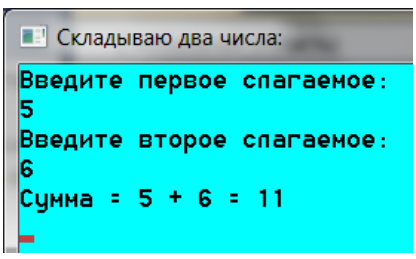

Рис. 3.2. Фрагмент работы консольного приложения

4. Итак, в данной программе Main() — это стартовая точка, с которой начинается ее выполнение. Обычно консольное приложение выполняется в окне на черном фоне. Чтобы как-то украсить традиционно черное окно консольного приложения, установим цвет фона окна BackgroundColor сине-зеленым (Cyan), а цвет символов, выводимых на консоль, черным (Black). Выводим строки в окно консоли методом WriteLine, а для считывания строки символов, вводимых пользователем, используем метод ReadLine.

5. Далее объявляем три переменных типа Single для соответственно первого числа, второго и значения суммы. Тип данных Single применяется тогда, когда число, записанное в переменную, может иметь целую и дробную части.

6. Переменная типа Single занимает 4 байта. Для преобразования строки символов, введенных пользователем в числовое значение, используем метод Parse.

7. После вычисления суммы необходимо вывести результат вычислений из оперативной памяти на экран. Для этого воспользуемся форматированным выводом в фигурных скобках метода WriteLine объекта Console:

Console::WriteLine("Сумма = {0} + {1} = {2}", X, Y, Z)

8. Затем выдаем звуковой сигнал Beep, символизирующий об окончании процедуры и выводе на экран результатов вычислений. Последняя строка в программе Console::ReadKey(); предназначена для приостановки выполнения программы до нажатия какой- нибудь клавиши. Если не добавить эту строку, окно с командной строкой сразу исчезнет, и пользователь не сможет увидеть вывод результатов выполнения. Программа написана.

9. Нажмите клавишу F5, чтобы увидеть результат.

Контрольные вопросы

1. Что такое форматированный ввод?
2. Какая команда помогает задерживать результат работы программы на экране?
3. Опишите вывод результата.

Приложение 1. Аннотация
рабочей программы дисциплины

АННОТАЦИЯ РАБОЧЕЙ ПРОГРАММЫ ДИСЦИПЛИНЫ

Дисциплина «Современные системы программирования» реализуется на факультете Информационных систем и безопасности кафедрой Информационных технологий и систем.

Цель дисциплины: получение базовых знаний о современных методах разработки прикладных программ и информационных систем на основе концепций объектно-ориентированного подхода, а также развитие навыков самостоятельной работы, связанных с анализом, детализацией, выбором методов решения поставленных задач, планированием использования возможностей современных сред программирования, а также различных источников информации для реализации программных приложений.

Задачи:

- получить представление об основных принципах процедурного программирования;
- получить представление об основных понятиях объектно-ориентированного проектирования и программирования;
- получить практические навыки по разработке программного обеспечения.

Дисциплина направлена на формирование следующих компетенций:
ПК-1. Способен проводить систематизацию, алгоритмизацию конкретных информационных потоков по месту научных исследований, производственной деятельности

В результате освоения дисциплины обучающийся должен:
Знать: способы описания алгоритма решения вычислительных задач, управляющие конструкции современных языков и правила оформления программного кода, структуру программы и базовые типы данных;
Уметь: работать с современными системами программирования, применять приемы и методы современного программирования, анализировать программу на предмет эффективности человеко-машинного взаимодействия и оптимальности программного решения;
Владеть: языками процедурного и объектно-ориентированного программирования, навыками разработки и отладки программ на языках высокого уровня, навыками оптимизации программного кода.

По дисциплине предусмотрена промежуточная аттестация в форме зачета.
Общая трудоемкость освоения дисциплины составляет 4 зачетные единицы.