МИНОБРНАУКИ РОССИИ

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

ИНСТИТУТ ИНФОРМАЦИОННЫХ НАУК И ТЕХНОЛОГИЙ БЕЗОПАСНОСТИ
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ СИСТЕМ И БЕЗОПАСНОСТИ
Кафедра информационных технологий и систем

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

10.03.01 Информационная безопасность
Код и наименование направления подготовки/специальности

Организация и технологии защиты информации
(по отрасли или в сфере профессиональной деятельности)
Наименование направленности (профиля)/ специализации

Уровень высшего образования: *бакалавриат*

Форма обучения: *очная*

РПД адаптирована для лиц
с ограниченными возможностями
здоровья и инвалидов

Москва 2026

Языки программирования
Рабочая программа дисциплины

Составитель:
Кандидат технических наук, доцент кафедры КЗИ А.С. Моляков

УТВЕРЖДЕНО
Протокол заседания кафедры
информационных технологий и систем
№ 5 от 11.12.2025

ОГЛАВЛЕНИЕ

1. Пояснительная записка

1.1. Цель и задачи дисциплины

Цель дисциплины*:* приобретение знаний, навыков и умений в области высокоуровневых языков программирования, а также освоение современных алгоритмов анализа больших данных.

Задачи дисциплины: изучение базовых принципов программирования; изучение специализированных технологий и методов программирования на языках C/C++ и Python для анализа и хранения данных; изучение главных управляющих структур языков при использовании функций Win API; приобретение навыков и умений по разработке алгоритмов в задачах анализа данных с использованием библиотек графической и потоковой обработки.

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

Компетенция (код и наименование)	**Индикаторы компетенций** (код и наименование)	**Результаты обучения**
УК-2 Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	УК-2.1 Способен применять соответствующий математический аппарат для решения профессиональных задач	Уметь: решать типовые программно-математические задачи защиты информации;
	УК-2.2 Способен использовать знания о важнейших нормах, институтах и отраслях действующего российского права для определения круга задач и оптимальных способов их решения	Владеть: навыками использования положений стандартов при разработке, настройке и оптимизации программных модулей на алгоритмических языках программирования
ОПК-7 Способен использовать языки программирования и технологии разработки программных средств для решения задач профессиональной деятельности	ОПК-7.1 Знает основные принципы построения компьютера, формы и способы представления данных; области и особенности применения языков программирования высокого уровня	Знать: основные принципы и способы представления данных; построения вычислительных блоков компьютерных систем; области и особенности применения языков программирования высокого уровня (C/C++/Phyton)
	ОПК-7.2 Умеет работать с интегрированной средой разработки программного обеспечения; работать с интегрированной средой разработки программного обеспечения; разрабатывать и реализовывать на языке высокого уровня алгоритмы решения типовых профессиональных задач	Уметь: работать с интегрированной средой разработки программного обеспечения; работать с интегрированной средой разработки и реализовывать алгоритмы на примере MS Visual Studio и Phyton Shell
	ОПК-7.3 Владеет навыками разработки,	Владеть: навыками разработки, документирования, тестирования и отладки

	документирования, тестирования и отладки программ; разработки алгоритмов решения типовых профессиональных задач	программ; разработки алгоритмов на примере MS Visual Studio и Phyton Shell

1.3. Место дисциплины в структуре образовательной программы

Дисциплина «Языки программирования» относится к обязательной части блока дисциплин учебного плана.

2. Структура дисциплины

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

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
2	Лекции	**28**
2	Практические занятия	**32**
2	Промежуточная аттестация	**18**
	Всего:	**60**

Объем дисциплины в форме самостоятельной работы обучающихся составляет 66 академических часа.

3. Содержание дисциплины

№	Наименование раздела дисциплины	Содержание
1	Введение в дисциплину	Общие положения. Базовые термины, понятия и определения. Классическая теория алгоритмов
2	Основы теории алгоритмических языков программирования	Синтаксис языков программирования. Операционная семантика Место компилятора в программном обеспечении. Машина Тьюринга. Принципы фон-Неймановской архитектуры
3	Основные принципы и способы представления данных; построения вычислительных блоков компьютерных систем	Критерии эффективности языков программирования. Простота прочтения и понимания программного языка. Простота реализации программ, то есть насколько удобен в применении язык для конкретной сферы. Критерий надёжности. Имеется в виду способность языка обеспечить минимум ошибок при выполнении программ. Общая себестоимость жизненного цикла языка программирования.
4	Интегрированные среды разработки программного обеспечения	Интегрированная среда разработки как система программных средств, используемая программистами для разработки программного обеспечения. Среда разработки включает в себя: текстовый редактор; компилятор и / или интерпретатор; средства автоматизации сборки; отладчик. Visual Studio 2003/2012/2019. Быстрое написание кода. Phyton shell. Автоматизация работы.

5	Программирование на C/C++/Phyton.Работа с массивами и строковыми переменными	Массивы и строковые переменные в языках C/C++/Phyton.Многомерные массивы Типы операндов. Форматы данных. Оптимизация работы IntelliSense в файлах C++. Локальная разработка с поддержкой множества популярных эмуляторов.
6	Программирование на C/C++/Phyton.Работа с математическими формулами	Руководство. Определение и использование классов и структур .Упрощенный доступ к тестам в обозревателе решений. Интерфейс Git для создания и клонирования репозиториев, управления ветвями
7	Программирование на C/C++/Phyton. Работа с графическими объектами	Отображение изображений с помощью .NET Framework. Рисование фигур с помощью .NET Framework. Вращение изображений с помощью .NET Framework. Преобразование форматов файлов изображений
8	Базовые навыки сборки и тестирования программных модулей	Компиляция программы на C/C++ включает взятие написанного нами исходного кода (файлы .cpp, .c, .h, .hpp) и преобразование их в исполняемый файл или библиотеку, которая может работать на указанной платформе. Этот процесс можно разделить на три основных этапа: Препроцесинг, Компиляция, Компоновка. Типичный рабочий процесс при создании программы - сборка, а затем отладка.

4. Образовательные технологии

№ п/п	**Наименование темы**	**Виды учебных занятий**	**Образовательные технологии**
1	**2**	**3**	**4**
1	Введение в дисциплину	Лекция, Опрос 1.1 Лекция, Опрос 1.2 Самостоятельная работа	Традиционная с использованием презентаций Тестирование Подготовка к занятиям с использованием ЭБС
2	Основы теории алгоритмических языков программирования	Лекция, Опрос 2.1 Лекция, Опрос 2.2 Практическое занятие 1. Самостоятельная работа	Лекция-дискуссия Тестирование Занятия с использованием специализированного ПО – MS Visual Studio и Phyton Shell Подготовка к занятиям с использованием ЭБС
3	Основные принципы и способы представления данных; построения вычислительных блоков компьютерных систем	Лекция, Опрос 3.1 Лекция, Опрос 3.2 Практическое занятие 2. Самостоятельная работа	Лекция-дискуссия Тестирование Занятия с использованием специализированного ПО – MS Visual Studio и Phyton Shell Подготовка к занятиям с использованием ЭБС
4	Интегрированные среды разработки программного	Лекция, Опрос 4.1 Лекция, Опрос 4.2	Проблемная лекция Традиционная с использованием

№ п/п	Наименование темы	Виды учебных занятий	Образовательные технологии
	обеспечения	Практические занятия 3. Самостоятельная работа	презентаций Тестирование Занятия с использованием специализированного ПО – MS Visual Studio и Phyton Shell Подготовка к занятиям с использованием ЭБС
5	Программирование на С/С++.Работа с массивами и строковыми переменными	Лекция, Опрос 5.1 Лекция, Опрос 5.2 Практическое занятие 4. Самостоятельная работа	Лекция с разбором конкретных ситуаций Тестирование Занятия с использованием специализированного ПО – MS Visual Studio и Phyton Shell Подготовка к занятиям с использованием ЭБС
6	Программирование на С/С++.Работа с математическими формулами	Лекция, Опрос 6.1 Лекция, Опрос 6.2 Практическое занятие 5. Самостоятельная работа	Лекция-дискуссия Тестирование Занятия с использованием специализированного ПО – MS Visual Studio и Phyton Shell Подготовка к занятиям с использованием ЭБС
7	Программирование на С/С++.Работа с графическими объектами	Лекция, Опрос 7.1 Лекция, Опрос 7.2 Практическое занятие 6. Самостоятельная работа	Лекция-дискуссия Тестирование Занятия с использованием специализированного ПО – MS Visual Studio и Phyton Shell Подготовка к занятиям с использованием ЭБС
8	Базовые навыки сборки и тестирования программных модулей	Лекция, Опрос 8.1 Лекция, Опрос 8.2 Самостоятельная работа	Лекция-дискуссия Тестирование Подготовка к занятиям с использованием ЭБС

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 (темы 1-8)* - *тестирование* - *практические задания (темы 2-4)* - *практические задания (темы 5-8)*	 2 балла 2 балла 4 балла 4 балла	 16 баллов 16 баллов 12 баллов 16 баллов
Промежуточная аттестация - *экзамен*		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	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	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 –

Баллы/ Шкала ECTS	Оценка по дисциплине	Критерии оценки результатов обучения по дисциплине
		«хороший».
67-50/ D,E	удовлетво-рительно	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	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

Контрольные вопросы

1. Введение. Переменные, константы. Типы данных. Операторы, выражения.
2. Ввод-вывод данных.
2. Библиотека Scikit-Learn.
3. Исключения C/C++/Python.
4. Ветвления и циклы в C/C++/Python.
5. Строки. Базовые операции. Функции и методы строк.
6. Списки. Базовые операции. Функции и методы списков. Генераторы списков. Вложенные списки.
7. Кортежи. Базовые операции.
8. Множества. Базовые операции. Генераторы множеств.
9. Словари. Создание, базовые операции. Методы словарей.
10. Функции. Передача параметров в функцию. Области видимости переменных.
11. Дополнительные возможности при работе с функциями.
12. MNIST.
13. Показатели производительности алгоритмов.
14. Классификация на множестве классов с использованием Python.
15. Принципы фон-Неймановской архитектуры.
16. Оптимизация на Python: пакетный градиентный спуск, стохастический градиентный спуск.
17. Обучение и визуализация дерева принятия решений.
18. Алгоритмы обучения . Понятие об энтропии.
19. Ансамблевое обучение. Бэггинг и вставка в Scikit-Leran.
20. Случайные леса на Python.
21. Framework TensorFlow. Создание графа.

22. Линейная регрессия с помощью TensorFlow.
23. Визуализация графа и кривых обучения с использованием TensorBoard.
24. Искусственные нейронные сети.
25. Персептрон и его многослойная модификация.

Примерные задания для тестирования

1. Что такое препроцессор?
- Составная системного блока, предназначенная для обработки данных
- Составная процессора, предназначенной для вычислений с плавающей запятой
+ Составляющая компиляции, которая обрабатывает директивы или команды

2. Что такое заголовочные файлы?
- Название программы, указывается при сохранении
- Название главной функции или функции пользователя
+ Модули, сохраняют заголовки функций
+ Стандартные библиотеки, расположенные в папке include

6. Учебно-методическое и информационное обеспечение дисциплины

6.1 Список источников и литературы

Литература
основная

1. Рояк, М. Э. Основы объектно ориентированного программирования на языке C#: базовые сведения о языке : учебное пособие / М. Э. Рояк, И. М. Ступаков. — Новосибирск : НГТУ, 2023. — 70 с. — ISBN 978-5-7782-5096-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/514402. — Режим доступа: для авториз. пользователей.
2. Информационные технологии и основы программирования : учебное пособие : в 2 частях / А. В. Петрищев, О. И. Лаптев, Т. В. Мятеж [и др.]. — Новосибирск : НГТУ, 2024 — Часть 1 — 2024. — 168 с. — ISBN 978-5-7782-5143-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/514506. — Режим доступа: для авториз. пользователей.
3. Информационные технологии и основы программирования : учебное пособие : в 2 частях. — Новосибирск : НГТУ, 2022 — Часть 2 — 2024. — 67 с. — ISBN 978-5-7782-5325-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/514509. — Режим доступа: для авториз. пользователей.
4. Колесникова, Т. Г. Языки программирования : учебное пособие / Т. Г. Колесникова. — Кемерово : КемГУ, 2019. — 182 с. — ISBN 978-5-8353-2448-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/134312. — Режим доступа: для авториз. пользователей.

Дополнительная

1. Токмаков, Г. П. Базы данных: Модели и структуры данных, язык SQL, программирование баз данных : учебное пособие / Г. П. Токмаков. — Ульяновск : УлГТУ, 2021. — 362 с. — ISBN 978-5-9795-2184-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/259706. — Режим доступа: для авториз. пользователей.

2. Алымова, Е. В. Конечные автоматы и формальные языки : учебник / Е. В. Алымова. В. М. Деундяк. А. М. Пеленицын ; Южный федеральный университет. - Ростов-на-Дону : Таганрог : Издательство Южного федерального университета. 2018. - 292 с. - ISBN 978-5-9275-2397-9. - Текст : электронный. - URL: https://znanium.com/catalog/product/1020503. – Режим доступа: по подписке.
3. Волк, В. К. Базы данных. Проектирование, программирование, управление и администрирование : учебник / В. К. Волк. – Санкт-Петербург : Лань, 2020. – 244 с. – ISBN 978-5-8114-4189-1. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/126933. – Режим доступа: для авториз. пользователей.
4. Сидорова, Н. П. Базы данных: практикум по проектированию реляционных баз данных : учебное пособие / Н. П. Сидорова. — Королёв : МГОТУ, 2020. — 92 с. — ISBN 978-5-4499-0799-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/149436. — Режим доступа: для авториз. пользователей.

6.2 Перечень ресурсов информационно-телекоммуникационной сети «Интернет».

Национальная электронная библиотека (НЭБ) www.rusneb.ru
ELibrary.ru Научная электронная библиотека www.elibrary.ru
Электронная библиотека Grebennikon.ru www.grebennikon.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:

1) для лекционных занятий - учебная аудитория, доска, компьютер или ноутбук, проектор (стационарный или переносной) для демонстрации учебных материалов.

Состав программного обеспечения:

1. Windows
2. Microsoft Office
3. Kaspersky Endpoint Security

2) для практических занятий – компьютерный класс или лаборатория, 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. Mozilla Firefox
5. Microsoft Share Point 2010
6. Vmware Player 15.5
7. Vmware Player 15.5
8. MS Visual Studio

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

Практическое занятие 1. Первоначальная настройка Microsoft Visual Studio Express

Выполнение задания :

1. Чтобы приступить к установке Microsoft Visual Studio Express, надо скачать дистрибутив с сайта Microsoft. Прокрутите страницу и найдите там блок с Express for Desktop.
2. Перед тем как нажать кнопку Загрузка, просмотрите пункты System Requirements (системные требования к вашему компьютеру).
3. Если они не отвечают вашим, например, у вас более старая версия Windows, то найдите в Google Microsoft Visual Studio 2010 Express или Microsoft Visual Studio 2013 Express и выберите ту версию, которая сможет нормально работать на вашем компьютере.
4. Если все требования выполняются – жмите Загрузка
5. После завершения установки настраиваем среду разработки. Студенты создают новый проект, внести в него код и запустить. В открывшейся MVS нажимаем – Создать проект.
6. В этом окне нажмите на Visual C++, Win32, Консольное приложение Win32, введите любое имя для вашего нового проекта и снимите галочку “Создать каталог для решения”. Жмем ОК.
7. Создаем файл с расширением .cpp. В него мы будем писать код программы. Делается это так: в окошке Обозреватель решений нажмите правой кнопкой мыши на имя вашего проекта (у меня это ConsoleApplication2)
8. После написания тестового кода обучающимися нажмите сочетание клавиш Ctrl + Shift + B – начнется компиляция программы. Вы должны увидеть в нижней строке окна Сборка: успешно: 1, с ошибками: 0 и т.д. После этого нажимаем Ctrl + F5 и видим в открывшемся окне сообщение: “Мы создали первый проект в MVS”

Замечания по выполнению задания:

1. переменным можно присвоить не только определённое значение, а и результат какого-то вычисления: amount_of_apples1 = amount_of_apples2 + 33;

2. объявлять переменные желательно в начале main-функции. А при необходимости еще и оставить комментарий, что они будут хранить.
3. регистр букв имени имеет значение. Имена Apple и apple обозначают разные переменные.

Практическое занятие 2. Работа со строками и константами

Выполнение задания:

1.Определить константу, которая будет хранить количество дней в неделе: const int daysInWeek = 7;

2.Чтобы дать понять компилятору, что это константа, а не обычная переменная, перед типом данных обязательно использовать ключевое слово const.

3. Чтобы показать значение переменной на экран, достаточно обратиться к ней по имени: cout << inBox; Обратите внимание, как работает cout.

4. С помощью оператора << , мы можем чередовать показ текста и показ значения переменной. Так мы делали в строках 17, 19, 20 и 27. В строке 24 в переменную inBox записывается сумма переменных: amount_of_apples1 + amount_of_apples2. То есть, компилятор, сначала сложит значения этих переменных, а потом запишет сумму в inBox .

5. Выводим значение inBox на экран – строка 27. Запускаем программу (сначала Ctrl + Shift + B, если ошибок нет то далее Ctrl + F5)

6. Выполнить тестовые задания на C/C++ и Phyton.

7. Подготовить Отчет.

Замечания по выполнению задания:

1 тип данных указывается перед именем переменной и определяет какие данные в ней будут храниться (число, символ…) и сколько памяти необходимо под них выделить.

2 имя переменной дает программист, соблюдая определённые правила (указаны выше).

3 чтобы создать (объявить) переменную надо указать её тип и дать имя. Переменную желательно сразу инициализировать (присвоить значение при создании): тип имя = значение;

4 чтобы объявить константу необходимо использовать ключевое слово const и обязательно сразу присвоить значение: const тип имя = значение.

Практическое занятие 3. Вывод данных на экран и ввод данных с клавиатуры

Выполнение задания:

1. Изучить несколько специальных символьных последовательностей, которые помогут нам манипулировать выводом данных на экран.
2. Вывести на экран цитату из фильма. Сразу пусть прозвучит сигнал, который привлечет внимание пользователя на экран.
3. По центру разместим заголовок и название фильма, из которого цитируем, а ниже – цитату.

4. Организовать ввод данных используя операторы cin и >> . Синтаксис следующий: cin >> имя Переменной.
5. Запустить программу и обратите внимание – программа выполнит команду 11-й строки и, дойдя до оператора cin >>, остановится и начнется ожидание действия от пользователя.
6. Надо ввести значение и нажать Enter. Как только переменная получит значение, введенное с клавиатуры, программа продолжит выполнение.
7. Научиться применять унарные операторы, для которых необходим один операнд. Называются они инкремент (++) и декремент (—). Роль этих операторов в том, чтобы изменить (увеличить или уменьшить соответственно) значение переменной на единицу, при этом значение будет перезаписано.
8. Рассмотреть пример, в котором будем изменять значение переменной variable на единицу тремя различными способами, как указано в тестовом примере.
9. После каждой операции в строках 11, 13 и 15. к значению переменной variable прибавляется единица. Как вы видите, самая короткая запись – это запись с использованием инкремента. Ниже, в строках 17 – 19, мы трижды применили декремент и в итоге получим значение variable уменьшенное на 3.
10. Выполнить тестовые задания на C/C++ и Phyton.
11. Подготовить Отчет.

Контрольные вопросы:

1. Математические функции стандартной библиотеки Си (<math.h>).
2. Форматированный ввод-вывод (<stdio.h>): параметры функций printf(), scanf().
3. Файловый ввод-вывод (<stdio.h>): форматированный и бесформатный.
4. Массивы. Передача массивов в функции.
5. Определение функции. Прототип функции. Рекурсия.

Практическое занятие 4. Генератор случайных чисел

Выполнение задания:

1. Если воспользоваться только функцией rand() – получите одинаковые “случайные числа” от запуска к запуску.
2. Набрать следующий код и откомпилируйте программу несколько раз. Обратите внимание, что “случайные числа” всегда будут одинаковы.
3. Случайное число генерируется в строке 11 и записывается в i-й элемент массива randomDigits.

4. Числа генерируются не совсем случайные. Чтобы добиться “настоящей” случайности чисел при повторных запусках программы, необходимо применить функцию srand() до функции rand(). При этом надо передать ей в виде параметра функцию time() с параметром NULL: srand(time(NULL)); (параметр или аргумент функции – это то, что прописывается в круглых скобках после имени функции
5. srand() получает в виде параметра текущее системное время, которое при каждом запуске программы будет разным. Это позволит функции rand() каждый раз генерировать именно случайные числа.
6. В первом цикле for происходит генерация случайных чисел определённых диапазонов и их запись в соответствующие массивы. В каждом шаге цикла будут генерироваться новые случайные числа. Возможно кому-то сложно разобраться как это происходит. Рассмотрим детально:
7. rand() % 7 – rand() генерирует число и далее вычисляется остаток от деления на 7 от этого числа. Понятно, что это могут быть числа только от 0 до 6. Например генерируется 50 – остаток от деления на 7 будет равен 1, генерируется 49 – остаток от деления на 7 будет равен 0.
8. Выполнить тестовые задания на C/C++ и Phyton.
9. Подготовить Отчет.

Замечание по выполнению задания:

1 1 + rand() % 7 – очень похоже на предыдущий случай, только 0 мы уже не увидим, а вот 7 появится в диапазоне. Например генерируется 49 – остаток от деления на 7 равен 0 и к нему добавляется единица, генерируется 6 – остаток от деления на 7 равен 6 и опять же добавляется единица.

2 200 + rand() % 101 – даст нам число от 200 до 300. Например генерируется 100 – остаток от деления на 101 равен 100 и добавляется 200. Получаем число 300. Генерируется 202: 200 + (202 % 101)= 200 + 0 = 200.

3 rand() % 41 - 20 – от – 20 до 20. Например генерируется 1: (1 % 40) – 20 = 1 – 20 = -19; генерируется 30: 30 – 20 = 10.

4 0,01* (rand() % 101) – от 0.01 до 1. Например генерируется 55: 0.01* 55 = 0.55.

5 Для использования time() необходимо подключить библиотечный файл ctime (time.h для более старых компиляторов)

Практическое занятие 5. Работа с таблицами. Одномерные и многомерные массивы

Выполнение задания:

1. Массив – это совокупность определенного количества однотипных переменных, имеющих одно имя. Например, int array [3];. Эта запись означает, что мы объявили массив с именем array , который содержит в себе 3 переменные типа int.
2. Переменные массива называют элементами
3. Каждый элемент имеет свой уникальный индекс – свой порядковый номер. Используя индекс, Вы можете обращаться к конкретному элементу.
4. В строке 12 определить целочисленную константу SIZE, которая будет хранить размер массива (определённое нами, количество его элементов).
5. В строке 13 объявить массив: указываем тип данных, которые будут храниться в ячейках массива, даем имя и указываем размер в квадратных скобках.
6. попробовать в нашем примере внести любую другую цифру в константу SIZE. И вы увидите, что программа будет прекрасно работать – создаст массив на столько элементов, на сколько вы укажете, внесет данные и отобразит их на экране.
7. В строках 15 – 19 определить цикл for. Его счетчик i будет служить индексом элементов массива. В самом начале, он равен 0 и с каждым шагом будет увеличиваться на единицу до тех пор, пока не станет равным SIZE – количеству элементов массива.
8. Изучить многомерные массивы данных на примере с паркингом. Показать пользователю схему паркинга: этажи и места для парковки. Чтобы забронировать место он должен выбрать номер этажа и номер места. После бронирования – записать значение 0 в соответствующую ячейку, что будет означать “место занято”.
9. Выполнить тестовые задания на C/C++ и Phyton..
10. Оформить Отчет.

Контрольные вопросы:

1. Массивы. Передача массивов в функции.
2. Массивы: одномерные и двумерные.
3. Модульный подход в программировании. Использование *.h файлов. Раздельная компиляция
4. Оператор typedef. Приведение типов.
5. Операторы в выражениях языка Си. Приоритет операторов. Оператор sizeof().
6. Операторы инкремента и декремента.

Практическое занятие 6. Циклы и операторы ветвления

Выполнение задания:

1.Рассмотреть операторы, которые применяются ниже для логических операций:

Операторы сравнения (операторы отношения)		
Оператор	как бы задаем вопрос	результат при сравнении значений 3 и 8
> (больше)	значение слева больше чем справа?	cout << (3 > 8); на экране 0 (false)
< (меньше)	значение слева меньше чем справа?	cout << (3 < 8); на экране 1 (true)
>= (больше или равно)	значение слева больше или равно значению справа?	cout << (3 > =8); на экране 0 (false)
<= (меньше или равно)	значение слева меньше или равно значению справа?	cout << (3 <= 8); на экране 1 (true)
Операторы равенства		
== (равно)	значение слева равно значению справа?	cout << (3 == 8); на экране 0 (false)
!= (не равно)	значение слева не равно значению справа?	cout << (3 != 8); на экране 1 (true)

1. Изучить синтаксис операторов if / if else. Понять принцип работы оператора выбора if – если условие в круглых скобках истина (true), то код блока выполнится. Если ложь (false) – игнорируется и выполнение программы начинается со строки кода, следующей под блоком if.
2. Разобраться с тем, что означает цикл в программировании. Цикл – это специальный оператор, с помощью которого происходит повторение определённого участка кода определённое количество раз (какой участок кода повторять и когда цикл должен прерваться – определяет программист)
3. Не обязательно использовать постфиксный инкремент или декремент. Изменяем управляющую переменную так, как того требует задача. Это может быть ++i, i += 2, i += 20, i -= 15…
4. Рассмотреть 4 тестовых примера, варианты которых раздает студентам преподаватель..
5. Управляющая переменная i изменяется от 8 до 88 включительно, при этом шаг изменения равен 8. То есть сначала i = 8, на второй итерации 16 и так далее до 88.
6. Управляющая переменная i изменяется от 3000 до 300 включительно, с уменьшением при каждой итерации на 300 (3000, 2700, 2400…)
7. Управляющая переменная i изменяется от 0 до 100 включительно, с увеличением при каждой итерации на 10. (0, 10, 20…100)
8. Управляющая переменная i изменяется от 3000 до 3 включительно, с делением при каждой итерации на 10. (3000, 300, 30, 3).
9. Оформить итоговый отчет о проделанной работе.

Контрольные вопросы:

1. Блоки и правила видимости переменных.
2. Виды операторов присваивания в языке Си.
3. Глобальные и внешние переменные.
4. Директива #define препроцессора и ее использование. Макроопределения с параметром.
5. Директива #include препроцессора и ее использование.
6. Директивы условной компиляции препроцессора и их использование.
7. Использование функций: заголовок, тело и вызов функции.
8. Логические (булевские) операторы и операторы сравнения.

Приложение 1. Аннотация
рабочей программы дисциплины

АННОТАЦИЯ РАБОЧЕЙ ПРОГРАММЫ ДИСЦИПЛИНЫ

Цель дисциплины*:* приобретение знаний, навыков и умений в области высокоуровневых языков программирования, а также освоение современных алгоритмов анализа больших данных.

Задачи дисциплины: изучение базовых принципов программирования; изучение специализированных технологий и методов программирования на языках C/C++ и Python для анализа и хранения данных; изучение главных управляющих структур языков при использовании функций Win API; приобретение навыков и умений по разработке алгоритмов в задачах анализа данных с использованием библиотек графической и потоковой обработки;

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

Знать: основные принципы и способы представления данных; построения вычислительных блоков компьютерных систем; области и особенности применения языков программирования высокого уровня (C/C++/Phyton)

Уметь: решать типовые программно-математические задачи защиты информации;работать с интегрированной средой разработки программного обеспечения; работать с интегрированной средой разработки и реализовывать алгоритмы на примере MS Visual Studio и Phyton Shell

Владеть: навыками использования положений стандартов при разработке, настройке и оптимизации программных модулей на алгоритмических языках программирования; навыками разработки, документирования, тестирования и отладки программ; разработки алгоритмов на примере MS Visual Studio и Phyton Shell