

УЧЕБНАЯ ПРОГРАММА  
по дисциплине  
**"ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С++"**

А.А.Быков

boombook@yandex.ru

Данный курс предназначен для слушателей математической интернет школы первого и второго годов обучения.

Основные цели данного курса:

- 1) Обеспечение слушателей инструментом для изучения курса «Компьютерное моделирование»ю
  - 2) Обеспечение подготовки к олимпиаде (экзамену) по специальности «информатика».
- На первом году изучаются главы 1–5, на втором году главы 6–10. В основу курса положен принцип практического освоения каждого изучаемого аспекта языка С++. Результат освоения каждой темы – работающий код, написанный слушателем в соответствии с заданием. Форма отчетности – зачет или экзамен по каждой главе. Зачет получает студент, выполнивший все практические задания. Экзамен состоит из двух частей, теоретической и практической. Теоретическая часть содержит ориентировочно 10 вопросов по всем изучаемым темам, на каждый из которых дается письменный ответ. Состав вопросов теоретической части по пособию [1]. Практическая часть состоит в создании проекта в среде MS VS С++ по заданию экзаменатора.
- Первая часть практического курса использует парадигму консольного программирования. Ввод и вывод осуществляется только через файл. Вторая часть курса использует интерактивное программирование на базе MFC.

Основная литература (доступна в виде документов pdf):

- [1] Дейтел Х., Дейтел П. Программирование на С++.  
✓ *Дейтел С++, §1.11,*
  - [2] Либерти Дж. Освой самостоятельно С++ за 21 день.  
✓ *Либерти С++21, глава 1.*
  - [3] Круглински Д., Уингоу С., Шефферд Дж.  
Программирование Microsoft Visual Studio.  
✓ *Круглински, Уингоу, Шефферд, глава 1.*
- Дополнительная литература:
- [4] Страуструп Б., Программирование на С++. Специальное издание.  
✓ *Страуструп С++, §1.11,*
  - [5] Янг М. Дж. Visual С++, полное руководство в 2-х т.  
✓ *Янг, Том I, глава 1,*
  - [6] Либерти Дж. Энциклопедия С++.  
✓ *Либерти С++ энциклопедия, глава 1.*

<b>ГЛАВА 1</b>	<b>ОСНОВНЫЕ ПОНЯТИЯ С++</b>	<b>7</b>
1.1	ПОНЯТИЕ MICROSOFT VISUAL STUDIO	7
1.1.1	Инсталляция MS Visual Studio 2005 и MS VC++2005	7
1.1.2	Среда программирования, компиляции и отладки. Понятие проекта	7
1.1.3	Проекты Console Application	7
1.1.4	Проекты win32	7
1.1.5	Проекты MFC. Особенности проектов Dialog based, SDI, MDI	7
1.2	ПОНЯТИЕ С++	7
1.2.1	Понятие: консольное приложение	8
1.2.2	Понятие: строка	8
1.2.3	Понятие: файл, имена и расширения	8
1.2.4	Понятие: вывод информации в файл	8
1.2.5	Понятие: Предопределенные объекты typedefs и typedefs	8
1.2.6	Понятие: манипулятор endl	8
1.2.7	Техника: Вывод в файл текстовой строки. Простейшее форматирование	8
1.3	ВСТРОЕННЫЕ ТИПЫ С++	9
1.3.1	Понятие объекта	9
1.3.2	Основные ключевые точки кода, связанные с каждым объектом	9
1.3.3	Понятие класса. Имя класса (типа)	9
1.3.4	Понятие встроенного типа (предопределенного системного класса)	9
1.3.5	Объект как представитель класса (типа)	10
1.3.6	Понятие имени объекта	10
1.3.7	Основные атрибуты объекта: поля данных и методы (функции)	10
1.3.8	Переменные и константные объекты (константы)	10
1.3.9	Понятие поля данных	10
1.3.10	Понятие функции. Основные действия, которые выполняет функция	10
1.3.11	Понятие конструирования и инициализации	10
1.4	АРИФМЕТИЧЕСКИЕ ТИПЫ	10
1.4.1	Арифметические типы	10
1.4.2	Типы int, unsigned int, long, unsigned long	10
1.4.3	Тип bool	10
1.4.4	Копирование значений переменных целого типа	10
1.4.5	Типы float, double	11
1.4.6	Арифметические операторы без преобразования типа	11
1.4.7	Типы char, char[], char*	12
1.4.8	Понятие длины (size). Функция sizeof(имя)	12
1.4.9	Переменные и константные объекты	12
1.4.10	Понятие преобразования типов	12
1.5	АРИФМЕТИЧЕСКИЕ ОПЕРАТОРЫ С++	12
1.5.1	Арифметические операции	12
1.5.2	Операция копирования	12
1.5.3	Операции сложения, умножения, вычитания, деления	12
1.5.4	Операция вычисления остатка от деления нацело	12
1.5.5	Операции инкремента ++ и декремента --	13
1.5.6	Операции << и >>	14
1.5.7	Префиксные и постфиксные операторы	14
1.5.8	Применение скобок	14
1.5.9	Понятие функции. Основные действия, которые выполняет функция	14
1.5.10	Стандартные арифметические операторы. Функция sqrt(double)	14
1.6	ПРЕОБРАЗОВАНИЕ АРИФМЕТИЧЕСКИХ ТИПОВ	15
1.6.1	Преобразование типов при выполнении арифметических операций	15
1.6.2	Преобразование int в double	15
1.6.3	Преобразование double в int. Функции double floor(double), double ceil(double)	15
1.6.4	Оператор typedef	16
1.7	ВЫВОД В ФАЙЛ ЗНАЧЕНИЙ АРИФМЕТИЧЕСКИХ ТИПОВ	16
1.7.1	Вывод int, double	16
1.7.2	Вывод с комментариями	16
1.8	ОПЕРАТОРЫ ЦИКЛА	16
1.8.1	Цикл for. Итератор (переменная цикла)	16
1.8.2	Инициализатор, условие, приращение	16
1.8.3	О значении итератора после выхода из цикла for	17

1.8.4	Применение счетчика числа итераций.	17
1.8.5	Оператор <i>break</i> .	17
1.8.6	Оператор <i>continue</i> .	17
1.8.7	Выход из цикла с помощью <i>goto</i> .	18
1.8.8	О надежности С++ кода: определение итератора внутри оператора <i>for</i> .	18
1.8.9	Применение цикла для вывода в файл текстовой строки.	18
1.8.10	Применение цикла для вывода в файл набора чисел.	18
1.8.11	Циклы с несколькими инициализаторами.	18
1.8.12	Циклы с несколькими операторами приращения.	19
1.8.13	Цикл <i>while</i> .	19
1.8.14	Цикл <i>while(true)</i> .	19
1.8.15	Цикл <i>do</i> .	20
1.8.16	Вложенные циклы.	20
1.9	ЛОГИЧЕСКИЕ ОПЕРАТОРЫ С++.	24
1.9.1	Бинарная булевская функция <i>==</i> для встроенных типов.	24
1.9.2	Логический бинарный оператор <i>  </i> .	24
1.9.3	Логический бинарный оператор <i>&amp;&amp;</i> .	24
1.9.4	Операторы <i>if</i> и <i>if-else</i> .	24
1.9.5	Оператор <i>switch</i> .	24
1.9.6	Совместное применение циклов и логических операторов.	24
1.10	ВВОД ТЕКСТА И ДАННЫХ ИЗ ФАЙЛА.	24
1.10.1	Предопределенный объект <i>ifstream</i> и его использование.	24
1.10.2	Ввод нескольких слов. Понятие буфера.	24
1.10.3	Понятие разделителя.	24
1.10.4	Ввод целых и вещественных значений.	24
1.11	ПРОСТОЙ ИНТЕРПРЕТАТОР.	24
1.11.1	Программирование интерпретатора значений и операций, вводимых из файла.	24
<b>ГЛАВА 2</b>	<b>ФУНКЦИИ</b>	<b>24</b>
1.12	ФУНКЦИИ.	24
1.12.1	Понятие функции.	24
1.12.2	Формальные параметры, возвращаемое значение.	24
1.12.3	Функция с пустым списком параметров.	24
1.12.4	Функция, не возвращающая значения.	24
1.12.5	Объявление (прототип) и описание функции.	24
1.12.6	Оператор вызова функции, фактические параметры.	24
1.12.7	Передача параметра в функцию по значению.	24
1.13	ПЕРЕДАЧА ПАРАМЕТРА ПО УКАЗАТЕЛЮ.	25
1.13.1	Передача входного параметра по указателю.	25
1.13.2	Передача выходного параметра по указателю.	25
1.13.3	Передача нескольких параметров по указателю.	25
1.13.4	Значение <i>NULL</i> .	25
1.13.5	Передача значения <i>NULL</i> для управления работой функции.	25
1.13.6	Значение параметра по умолчанию. Применение сравнения указателя с <i>NULL</i> .	25
1.14	ПЕРЕДАЧА ПАРАМЕТРА В ФУНКЦИЮ ПО ССЫЛКЕ.	25
1.14.1	Понятие ссылки. Описание, определение и инициализация.	25
1.14.2	Понятие типа ссылки.	25
1.14.3	Копирование ссылки.	25
1.14.4	Копирование ссылаемого объекта.	25
1.15	ЗАЩИТА ОПЕРАНДОВ	25
1.15.1	Константные функции и константные параметры.	26
1.16	ПАРАМЕТРЫ ПО УМОЛЧАНИЮ.	26
1.17	ПЕРЕГРУЖЕННЫЕ ФУНКЦИИ.	26
1.18	РЕКУРСИВНЫЕ ФУНКЦИИ.	26
1.18.1	Рекурсивные функции.	26
<b>ГЛАВА 3</b>	<b>МАССИВЫ И УКАЗАТЕЛИ</b>	<b>26</b>
1.19	УКАЗАТЕЛИ.	26
1.19.1	Понятие указателя. Описание, определение и инициализация.	26
1.19.2	Адрес объекта и указатель на объект.	26
1.19.3	Понятие типа указателя.	26
1.19.4	Операция разыменовывания указателя.	26
1.19.5	Копирование указателя.	26

1.19.6	Копирование указываемого объекта.....	26
1.19.7	Инкремент и декремент указателя.....	26
1.19.8	Константные указатели и указатели на константные объекты.....	26
1.19.9	Указатель на указатель.....	26
1.20	ОДНОМЕРНЫЕ МАССИВЫ.....	27
1.20.1	Понятие одномерного массива с фиксированной размерностью.....	27
1.20.2	Доступ к элементам одномерного массива с помощью переменной с индексом.....	27
1.20.3	Инициализация одномерного массива.....	27
1.20.4	Неявное указание размера массива при инициализации.....	27
1.20.5	Вывод значения массива в файл.....	27
1.21	ОДНОМЕРНЫЕ МАССИВЫ И УКАЗАТЕЛИ.....	27
1.21.1	Операции над указателями.....	27
1.21.2	Операции инкремента и декремента.....	27
1.21.3	Доступ к элементам одномерного массива с помощью указателей.....	27
1.21.4	Взаимосвязь указателей и массивов.....	27
1.22	ПЕРЕДАЧА ОДНОМЕРНОГО МАССИВА В ФУНКЦИЮ.....	27
1.23	ОПЕРАТОРЫ NEW И DELETE.....	27
1.23.1	Применение операторов new и delete для создания переменной.....	27
1.23.2	Применение операторов new и delete для создания одномерного массива.....	27
1.23.3	Методика проверки корректности операции выделения памяти.....	28
1.23.4	Понятие утечки памяти и способы диагностики.....	28
1.24	ОПЕРАЦИИ С МАССИВАМИ.....	28
1.24.1	Копирование массива.....	28
1.24.2	Сортировка массива.....	28
1.24.3	Константные строки.....	28
1.24.4	Массивы символов char[]="..."; Символ конца строки.....	28
1.24.5	Строка заданной длины.....	28
1.24.6	Буфер.....	28
1.24.7	Строка переменной длины.....	28
1.24.8	Функции копирования, конкатенации, усечения.....	29
1.24.9	Запрос длины строки.....	29
1.24.10	Доступ к отдельным символам строки.....	29
1.25	ДВУМЕРНЫЕ МАССИВЫ.....	29
1.25.1	Массивы указателей.....	29
1.25.2	Понятие двумерного массива с фиксированной размерностью.....	29
1.25.3	Доступ к элементам двумерного массива.....	29
1.25.4	Вывод двумерного массива в файл.....	29
1.26	ОПЕРАТОРЫ NEW И DELETE ДЛЯ ДВУМЕРНЫХ МАССИВОВ.....	29
1.26.1	Применение операторов new и delete для создания двумерного массива.....	30
1.27	ПЕРЕДАЧА ДВУМЕРНОГО МАССИВА В ФУНКЦИЮ.....	30
1.28	УКАЗАТЕЛИ НА ФУНКЦИИ.....	30
1.28.1	Указатели на функции.....	30
1.28.2	Массив указателей на функции.....	30
<b>ГЛАВА 4</b>	<b>ВВОД И ВЫВОД.....</b>	<b>30</b>
1.29	МАНИПУЛЯТОРЫ ПРИ ВЫВОДЕ.....	30
1.29.1	Установка ширины поля вывода.....	30
1.29.2	Установка точности.....	30
1.29.3	Обзор манипуляторов.....	30
1.30	МАНИПУЛЯТОРЫ ПРИ ВВОДЕ.....	30
1.30.1	Ввод встроенных типов.....	30
1.30.2	Функции распознавания образов.....	30
1.31	СОЗДАНИЕ ПРОЕКТА С НЕСКОЛЬКИМИ ФАЙЛАМИ.....	31
1.31.1	Основные и заголовочные файлы.....	31
1.31.2	Область видимости объекта.....	31
1.31.3	Время жизни объекта.....	31
1.31.4	Блок.....	31
1.31.5	Обеспечение доступа к объекту из нескольких файлов.....	31
1.31.6	Пять ключевых точек кода.....	31
<b>ГЛАВА 5</b>	<b>ОБЪЕКТНО ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА С++.....</b>	<b>31</b>
1.32	ПОНЯТИЕ КЛАССА.....	31
1.32.1	Понятие класса. Встроенные классы и классы пользователя.....	31

1.32.2	Протокол класса.....	31
1.32.3	Данные-члены (поля) и функции-члены.....	31
1.32.4	Открытые и закрытые члены класса.....	31
1.32.5	Дружественные функции.....	31
1.33	РАБОТА С ОБЪЕКТАМИ.....	31
1.33.1	Объявление и описание объекта данного класса.....	31
1.33.2	Доступ к данным и функциям для объектов и указателей на объекты.....	31
1.33.3	Соглашение об именах классов, объектов и указателей.....	31
1.34	СПОСОБЫ ДОСТУПА К ПОЛЯМ ДАННЫХ.....	32
1.34.1	Способы доступа к членам класса. Интерфейсные функции.....	32
1.34.2	Интерфейсные функции типа <i>get(...)</i> .....	32
1.34.3	Интерфейсные функции типа <i>set(...)</i> .....	32
1.34.4	Методы валидации параметров.....	32
1.34.5	Выдача протокола работы программы в файл.....	32
1.35	ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ.....	32
1.35.1	Константные объекты и константные функции-члены.....	32
1.35.2	Статические данные-члены и статические функции-члены.....	32
1.36	КОНСТРУКТОРЫ И ДЕСТРУКТОРЫ.....	32
1.36.1	Конструирование объекта.....	32
1.36.2	Принципы работы с указателями на объект класса пользователя.....	32
1.36.3	Инициализирующий конструктор (конструктор с параметрами).....	32
1.36.4	Конструктор с аргументами по умолчанию. Перегрузка конструктора.....	32
1.37	КОНСТРУИРОВАНИЕ ОБЪЕКТОВ, ВКЛЮЧАЮЩИХ МАССИВЫ.....	32
1.37.1	Конструирование и разрушение объектов, включающих поля данных – массивы с постоянной размерностью.....	32
1.37.2	Конструирование и разрушение объектов, включающих поля данных – массивы с переменной размерностью.....	32
1.38	КОПИРОВАНИЕ ОБЪЕКТОВ.....	32
1.38.1	Применение указателя <i>this</i> .....	33
1.38.2	Перегрузка оператора копирования <i>=</i> .....	33
1.38.3	Операция <i>swap(..., ...)</i> .....	33
1.38.4	Конструктор копирования.....	33
1.39	МАССИВЫ ОБЪЕКТОВ КЛАССА.....	33
1.39.1	Массив фиксированного размера объектов класса.....	33
1.39.2	Доступ к элементам массива.....	33
1.39.3	Массив переменного размера объектов класса.....	33
1.40	МАССИВ – ПОЛЕ ДАННЫХ КЛАССА ПОЛЬЗОВАТЕЛЯ.....	33
1.40.1	Массив фиксированного размера – поле данных класса.....	33
1.40.2	Массив переменного размера – поле данных класса.....	33
1.41	ПЕРЕГРУЗКА УНАРНЫХ ОПЕРАЦИЙ.....	33
1.41.1	Перегрузка операции сравнения.....	33
1.41.2	Перегрузка операции сравнения, пример: сортировка.....	33
1.41.3	Перегрузка унарных операций.....	33
1.42	ПЕРЕГРУЗКА БИНАРНЫХ ОПЕРАЦИЙ.....	33
1.42.1	Перегрузка бинарных операций для двух операндов одного типа.....	34
1.42.2	Перегрузка бинарных операций для операндов смешанного типа.....	34
1.43	ПРЕОБРАЗОВАНИЕ ТИПОВ.....	34
1.43.1	Преобразование типов для объектов встроенных классов.....	34
1.43.2	Перегрузка операций преобразования типов.....	34
1.44	ПЕРЕГРУЗКА ОПЕРАЦИИ ПОМЕЩЕНИЯ В ПОТОК И ВЗЯТИЯ ИЗ ПОТОКА.....	34
1.44.1	Перегрузка операции помещения в поток и взятия из потока.....	34
1.45	ПРИМЕРЫ КЛАССОВ, ОБЛАДАЮЩИХ БОЛЬШОЙ СТЕПЕНЬЮ ФУНКЦИОНАЛЬНОСТИ.....	34
1.45.1	Пример: класс <i>array</i> .....	34
1.45.2	Пример: класс <i>complex</i> .....	34
1.45.3	Пример: класс <i>string</i> для обработки строк.....	34
<b>ГЛАВА 6</b>	<b>ПРОИЗВОДНЫЕ КЛАССЫ.....</b>	<b>35</b>
1.46	ПОНЯТИЕ КЛАССА-КОНТЕЙНЕРА.....	35
1.46.1	Принципы наследования: контейнеризация (классы-контейнеры).....	35
1.47	ПОНЯТИЕ КЛАССА-ИТЕРАТОРА. НАСЛЕДОВАНИЕ.....	35
1.47.1	Принципы наследования: классы-итераторы.....	35
1.47.2	Статические поля данных в базовом классе.....	35
1.48	ПОНЯТИЕ УРОВНЯ ЗАЩИТЫ ПОЛЯ ДАННЫХ И ФУНКЦИИ (МЕТОДА) КЛАССА.....	36

1.48.1	Доступ к полям данных и методам базовых классов из объекта производного класса. Открытые, защищенные, закрытые данные-члены.	36
1.48.2	Открытые, защищенные, закрытые базовые классы.	36
1.49	РЕАЛИЗАЦИЯ ДОСТУПА К ПОЛЯМ ДАННЫХ И МЕТОДАМ ИЗ ДРУГИХ КЛАССОВ.	36
1.49.1	Дружественные функции и дружественные классы.	36
1.49.2	Приведение типов указателей базовых классов к указателям производных классов.	36
1.49.3	Переопределение функций-членов в производных классах.	36
1.50	МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ.	36
1.50.1	Классы, представляемые простым графом типа дерево.	36
1.50.2	Виртуальные базовые классы.	37
1.50.3	Виртуальные базовые классы.	37
1.51	КОНСТРУИРОВАНИЕ И РАЗРУШЕНИЕ СЛОЖНЫХ ОБЪЕКТОВ.	37
1.51.1	Порядок конструирования при одиночном наследовании.	37
1.51.2	Порядок конструирования при множественном наследовании.	37
<b>ГЛАВА 7</b>	<b>ВИРТУАЛЬНЫЕ ФУНКЦИИ И ПОЛИМОРФИЗМ</b>	<b>37</b>
1.52	ВИРТУАЛЬНЫЕ ФУНКЦИИ И ПОЛИМОРФИЗМ.	37
1.52.1	Виртуальные функции.	37
1.52.2	Виртуальные деструкторы.	37
1.52.3	Абстрактные классы.	37
1.52.4	Выполняемые абстрактные (чистые) виртуальные функции.	38
1.52.5	Иерархия абстрактных классов.	38
1.52.6	Раннее и позднее связывание.	38
1.53	ПОТОКИ ВВОДА-ВЫВОДА.	38
1.53.1	Понятие файлового потока вывода.	38
1.53.2	Вывод в файловый поток протокола работы программы.	38
1.53.3	Вывод в файловый поток данных класса пользователя.	38
1.53.4	Манипуляторы потока.	38
1.53.5	Форматирование при выводе.	38
1.53.6	Обработка ошибок.	38
1.53.7	Форматированный ввод из файла.	38
1.54	ШАБЛОНЫ.	39
1.54.1	Шаблоны классов.	39
1.54.2	Шаблоны функций.	39
1.55	РАБОТА С ФАЙЛАМИ.	39
1.55.1	Открытие и закрытие файлов.	39
1.55.2	Файлы последовательного доступа.	39
1.55.3	Файлы произвольного доступа.	39
1.56	ОБРАБОТКА ИСКЛЮЧЕНИЙ.	39
<b>ГЛАВА 8</b>	<b>БИБЛИОТЕКА ШАБЛОНОВ</b>	<b>39</b>
1.57	STL (STANDARD TEMPLATE LIBRARY), АБСТРАКЦИЯ ДАННЫХ.	39
1.57.1	Абстрактный тип данных «vector», реализация шаблона «vector» с параметром <int>.	39
1.57.2	Абстрактный тип данных «vector», реализация шаблона с параметром пользовательского типа <SMuTime>.	39
1.57.3	Абстрактный тип данных «deque», реализация шаблона с параметром пользовательского типа <SMuTime>.	39
1.57.4	Абстрактный тип данных «list», реализация шаблона с параметром пользовательского типа <SMuTime>.	40
1.57.5	Абстрактный тип данных «стек».	40
<b>ГЛАВА 9</b>	<b>ГРАФИЧЕСКИЙ ИНТЕРФЕЙС</b>	<b>40</b>
1.58	ПРОГРАММНАЯ ОБОЛОЧКА ДЛЯ ИЗУЧЕНИЯ ОБЪЕКТНО ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ С++.	40
1.58.1	Dialog based application.	40
1.58.2	Элемент управления типа Static Text.	40
1.58.3	Элемент управления Button, button RUN, изменение значений переменных в функции OnRun().	40
1.58.4	Создание статических элементов управления и редактируемых элементов управления.	41
1.58.5	Инициализация переменных в функции InitDialog().	41
1.58.6	Чтение информации с диалоговой панели.	41
1.58.7	Отображение информации на диалоговой панели.	41
<b>ГЛАВА 10</b>	<b>ПРОГРАММИРОВАНИЕ MFC</b>	<b>41</b>

1.59	МЕТОДИКА ВСТРАИВАНИЯ КЛАССА ПОЛЬЗОВАТЕЛЯ В MFC ПРИЛОЖЕНИЕ. ....	41
1.60	ОСНОВНЫЕ ОПЕРАТОРЫ ВЫВОДА ИНФОРМАЦИИ. ....	41
1.60.1	<i>Редактор ресурсов. Понятие Static box и Edit box.</i> .....	41
1.60.2	<i>Простейший набор операторов форматирования.</i> .....	41
1.60.3	<i>Элементарные операции со строками.</i> .....	41
1.60.4	<i>Операторы графического вывода. Рисование линий и простейших форм.</i> .....	41
1.60.5	<i>Методика встраивания класса пользователя в MFC приложение, 1.</i> .....	42
1.60.6	<i>Методика встраивания класса пользователя в MFC приложение, 2.</i> .....	42
1.61	МЕТОДИКА ВСТРАИВАНИЯ КЛАССА ПОЛЬЗОВАТЕЛЯ В MFC ПРИЛОЖЕНИЕ. ....	42
1.62	ИНИЦИАЛИЗАЦИЯ, УПРАВЛЕНИЕ ДАННЫМИ, ОПЕРАТОРЫ ИСПОЛНЕНИЯ. ....	42
1.63	КОНСТРУИРОВАНИЕ ДИАЛОГОВОЙ ПАНЕЛИ. ....	42
1.64	СТАНДАРТНЫЙ ДИАЛОГ РАБОТЫ С ФАЙЛАМИ. ....	42
1.65	УПРАВЛЕНИЯ РАБОТОЙ МНОГОПОТОЧНОГО ПРИЛОЖЕНИЯ. СЕМАФОРЫ И Т.Д. ....	42
1.66	ГРАФИЧЕСКОЕ ОТОБРАЖЕНИЕ ИНФОРМАЦИИ.....	42
1.66.1	<i>Программирование простейшего графического класса.</i> .....	42
1.66.2	<i>Программирование двумерной графики.</i> .....	42
1.66.3	<i>Программирование квазитрехмерной графики.</i> .....	42
1.67	ПРОГРАММИРОВАНИЕ ТАЙМЕРА. ....	42
1.68	МНОГОПОТОЧНЫЕ ПРИЛОЖЕНИЯ. ПРОГРАММИРОВАНИЕ ИНТЕРФЕЙСНОГО ПОТОКА. ....	42
1.69	МНОГОПОТОЧНЫЕ ПРИЛОЖЕНИЯ. ПРОГРАММИРОВАНИЕ РАБОЧИХ ПОТОКОВ. ....	42
<b>ГЛАВА 11 ПРОГРАММИРОВАНИЕ SDI И MDI ПРИЛОЖЕНИЙ.....</b>		<b>42</b>
1.70	АРХИТЕКТУРА ДОКУМЕНТ-ВИД.....	43
1.71	СОЗДАНИЕ SDI ПРИЛОЖЕНИЙ.....	43
1.71.1	<i>Программирование простейшего SDI приложения с элементами отображения.</i> .....	43
1.71.2	<i>Программирование SDI приложения с диалогом.</i> .....	43
1.72	РЕАЛИЗАЦИЯ ПРЕДСТАВЛЕНИЯ ДАННЫХ. ....	43
1.73	РЕАЛИЗАЦИЯ ДОКУМЕНТА. ....	43
1.74	СЕРИАЛИЗАЦИЯ. ХРАНЕНИЕ ДОКУМЕНТА НА ВНЕШНЕМ НОСИТЕЛЕ. ....	43
1.74.1	<i>Понятие сериализации.</i> .....	43
1.74.2	<i>Применение сериализации для сохранения объектов класса пользователя.</i> .....	43
1.75	ПАНЕЛИ ИНСТРУМЕНТОВ И СТРОКА СОСТОЯНИЯ. ....	43
1.76	ПЕРЕМЕЩАЕМЫЕ ПАНЕЛИ. ....	43
1.77	СОЗДАНИЕ MDI ПРИЛОЖЕНИЙ. ....	43
1.78	МНОГОПОТОЧНОСТЬ В SDI И MDI ПРИЛОЖЕНИЯХ. ....	43
1.79	ПОНЯТИЕ OLE. ....	43
1.80	ПОНЯТИЕ ACTIVE X. ....	43

## Глава 1 Основные понятия С++

### Лекция 1. Основные понятия С++ и Microsoft Visual Studio

#### 1.1 Понятие Microsoft Visual studio.

- ✓ Круглински, Уингоу, Шефферд, глава 1.
- ✓ Янг, Том 1, глава 1, 2,
- ✓ Дейтел С++, §1.15,

##### 1.1.1 Инсталляция MS Visual Studio 2005 и MS VC++2005.

##### 1.1.2 Среда программирования, компиляции и отладки. Понятие проекта.

##### 1.1.3 Проекты Console Application.

##### 1.1.4 Проекты win32.

##### 1.1.5 Проекты MFC. Особенности проектов Dialog based, SDI, MDI.

#### Практика 1. Инсталляция продукта MS-VS-2008.

Задание. Инсталлируйте Microsoft Visual Studio 2008.

#### 1.2 Понятие С++

- ✓ Страуструп С++, §3.2.
- ✓ Дейтел С++, §1.14 – 1.16,
- ✓ Либерти С++21, глава 2.

1.2.1 Понятие: консольное приложение.

1.2.2 Понятие: строка.

1.2.3 Понятие: файл, имена и расширения.

1.2.4 Понятие: вывод информации в файл.

1.2.5 Понятие: Предопределенные объекты `myofs` и `myifs`.

1.2.6 Понятие: манипулятор `endl`.

1.2.7 Техника: Вывод в файл текстовой строки. Простейшее форматирование.

### Практика 2. x01n01- Вывод текста в файл.

Листинг кода (файл `my.cpp`)

// Первая часть кода, обеспечивает вывод информации в файл:

```
// my.cpp : Defines the entry point for the console application.
```

```
//  
#include <iostream>  
#include <fstream>  
#include <iomanip>  
#include <math.h>  
//  
using namespace std;  
using std::ofstream;  
using std::ifstream;  
using std::cout;  
using std::cin;  
using std::setw;  
using std::endl;  
//  
ofstream myofs;  
ifstream myifs;  
//
```

// Функция `main()`, обеспечивает исполнение кода:

Листинг кода (файл `my.cpp`)

```
int main()  
{  
// x01n01. Откроем файл протокола для записи и поставим подпись:  
myofs.open("myproto.txt");  
myofs << "x01n01. Откроем файл протокола для записи и поставим подпись." << endl;  
myofs << "Иванов Иван Иванович, 18 февраля 2009 г." << endl;  
myofs << "Практическое задание x01n01. Простейшая программа С++" << endl << endl;  
// Закроем файл протокола:  
myofs.close();  
return 0;  
}
```

(Конец листинга)

Результат работы кода, файл `myproto.txt`:

```
x01n01. Откроем файл протокола для записи и поставим подпись.
```

```
Иванов Иван Иванович, 18 февраля 2009 г.
```

```
Практическое задание x01n01. Простейшая программа С++
```

```
Hello, world!
```

(конец вывода результата работы кода)

**Задание.** Измените код так, чтобы в начале отчета располагалась информация о Вас (Фамилия, имя отчество, номер учебной группы, номер задания, название задания, дата выполнения). В дальнейшем каждый отчет оформляйте аналогичным образом, указывая текущий номер задания и тему работы.

**Задание.** Экспериментально определите значение манипулятора endl.

**Задание.** Что получится, если в конце оператора не поставить символ ;

### **Практика 3. x01n02- Операции с константами.**

Функция main(), обеспечивает исполнение кода:

**Листинг кода (файл my.cpp)**

```
{
  myofs.open("myproto.txt", ios_base::app);
  myofs << "x01n02. Арифметические операции с константами типа int и double " <<
  endl;
  myofs << "Попробуем вывести константы в файл: " << endl;
  myofs << " 7=" << 7 << endl;
  myofs << " 7.0=" << 7.0 << endl;
  myofs << " 7/3=" << 7/3 << endl;
  myofs << " 7.0/3.0=" << 7.0/3.0 << endl;
  myofs << " (7+6)/3=" << (7+6)/3 << endl;
  myofs << " (7+6)/3.0=" << (7+6)/3.0 << endl;
  myofs << " (7.0+6)/3=" << (7.0+6)/3 << endl;
  myofs.close();
}
```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

x01n02. Арифметические операции с константами типа int и double

Попробуем вывести константы в файл:

```
7=7
7.0=7
7/3=2
7.0/3.0=2.33333
(7+6)/3=4
(7+6)/3.0=4.33333
(7.0+6)/3=4.33333
```

**(конец вывода результата работы кода)**

**Задание.** Объясните разницу результатов выполнения операций с числовыми константами с фиксированной запятой (целочисленными) и с числовыми константами с плавающей запятой (вещественными, более точно, рациональными).

**Задание.** Экспериментально определите значение опции ios\_base::app.

## **Лекция 2. Встроенные типы C++.**

### **1.3 Встроенные типы C++.**

✓ *Страуструп C++, §4.1-4.6.*

✓ *Либерти C++21, глава 3.*

✓ *Дейтел C++, §1.20,*

#### **1.3.1 Понятие объекта.**

#### **1.3.2 Основные ключевые точки кода, связанные с каждым объектом.**

- A) Объявление,
- B) Определение,
- C) Конструирование,
- D) Инициализация,
- E) Использование,
- F) Разрушение.

#### **1.3.3 Понятие класса. Имя класса (типа).**

#### **1.3.4 Понятие встроенного типа (предопределенного системного класса).**

**1.3.5 Объект как представитель класса (типа).****1.3.6 Понятие имени объекта.****1.3.7 Основные атрибуты объекта: поля данных и методы (функции).****1.3.8 Переменные и константные объекты (константы).****1.3.9 Понятие поля данных.****1.3.10 Понятие функции. Основные действия, которые выполняет функция:**

- G) Изменение значения полей данных объекта,
- H) Изменение порядка исполнения операторов (функций),
- I) Изменение состояния внешних устройств.

**1.3.11 Понятие конструирования и инициализации.****1.4 Арифметические типы.**✓ *Страуструп С++, §4.4-4.5.*✓ *Дейтел С++, §1.18,*✓ *Либерти С++21, глава 3.***1.4.1 Арифметические типы.****1.4.2 Типы int, unsigned int, long, unsigned long.****1.4.3 Тип bool.****Практика 4. x02n01. Операции с переменными типа int.**Функция `main()`, обеспечивает выполнение кода:Листинг кода (файл `my.cpp`)

```

{
    myofs << "x02n01. Конструирование переменных целого типа, " << endl;
    myofs << "инициализация переменных целого типа, " << endl;
    myofs << "присваивание значений переменной целого типа." << endl;
    myofs << "Создадим переменную m типа int, но инициализировать не будем." << endl;
    myofs << "Создадим и инициализируем переменную n типа int." << endl;
    myofs << "Создадим и инициализируем переменную k типа int." << endl;
    int m;
    int n=17;
    int k=12345678901;
    myofs << " m=" << m << endl;
    myofs << " n=" << n << endl;
    myofs << " k=" << k << endl;
    myofs << " Ошибка: Переменная m используется раньше, чем ей присвоено значение," <<
endl;
    myofs << " Ошибка: Попытка присвоить переменной k значение, выходящее за допустимые
границы." << endl << endl;
}

```

**(конец листинга)**Результат работы кода, файл `myproto.txt`:

```

x02n01. Инициализация переменных целого типа
m=-858993460
n=17
k=-539222987
Ошибка: Переменная m используется раньше, чем ей присвоено значение,
Ошибка: Попытка присвоить переменной k значение, выходящее за допустимые границы.

```

**(конец вывода результата работы кода)**

**Задание.** Объясните две диагностические выдачи на этапе исполнения. Исправьте код так чтобы все операции стали корректными.

**1.4.4 Копирование значений переменных целого типа.****Практика 5. x02n02. Копирование переменных типа int.**Функция `main()`, обеспечивает выполнение кода:Листинг кода (файл `my.cpp`)

```

{

```

```

myofs << "x02n02. Копирование переменных целого типа" << endl;
myofs << " Переменные m, n, k не инициализированы, но им присвоены значения" <<
endl;
int m, n, k, p;
m=123;
n=-789;
k=987654;
p=n;
myofs << " m=" << m << " n=" << n << " k=" << k << " p=" << p << endl << endl;
}

```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

```

x02n02. Копирование переменных целого типа
Переменные m, n, k не инициализированы, но им присвоены значения
m=123 n=-789 k=987654 p=-789

```

**(конец вывода результата работы кода)**

**Задание.** Создайте несколько переменных, используйте операцию копирования.

### 1.4.5 Типы float, double.

#### **Практика 6. x02n03. Инициализация переменных типа double.**

Функция main(), обеспечивает исполнение кода:

**Листинг кода (файл my.cpp)**

```

{
myofs << "x02n03. Инициализация переменных типа double" << endl;
double x, y=123.567, z=3.45e123;
myofs << " x=" << x << endl << " y=" << y << endl << " z=" << z << endl;
myofs << " Ошибка: Переменная x используется раньше, чем ей присвоено значение," <<
endl;
myofs << " Ошибка: Попытка присвоить переменной z значение, выходящее за допустимые
границы." << endl << endl;
}

```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

```

x02n03. Инициализация переменных типа double
x=-9.25596e+061
y=123.567
z=3.45e+123
Ошибка: Переменная x используется раньше, чем ей присвоено значение,
Ошибка: Попытка присвоить переменной z значение, выходящее за допустимые границы.

```

**(конец вывода результата работы кода)**

**Задание.** Разберитесь в значении фигурных скобок, обрамляющих код каждого упражнения.

### 1.4.6 Арифметические операторы без преобразования типа.

#### **Практика 7. x02n04. Арифметические операции без преобразования типов.**

**Листинг кода (файл my.cpp)**

```

{
myofs << "x02n04. Арифметические операции без преобразования типов" << endl;
int m=47, n=8;
myofs << " int m=" << m << ", int n=" << n << ", m+n=" << m+n << ", m*n=" << m*n <<
", m/n=" << m/n << endl;
double x=47, y=8;
myofs << " double x=" << x << ", double y=" << y << ", x+y=" << x+y << ", x*y=" <<
x*y << ", x/y=" << x/y << endl << endl;
}

```

**(конец листинга)**

Результат работы кода, файл myproto.txt:

```
x02n04. Арифметические операции без преобразования типов
int m=47, int n=8, m+n=55, m*n=376, m/n=5
double x=47, double y=8, x+y=55, x*y=376, x/y=5.875
```

(конец вывода результата работы кода)

*Задание.* См. сборник задач.

### 1.4.7 Типы char, char[], char\*.

### 1.4.8 Понятие длины (size). Функция sizeof(имя).

### 1.4.9 Переменные и константные объекты.

### 1.4.10 Понятие преобразования типов.

## **Практика 8. x02n05. Арифметические операции с явным преобразованием типов.**

Листинг кода (файл my.cpp)

```
{
myofs << "x02n05. Арифметические операции с явным преобразованием типов" << endl;
int m=47, n=8;
myofs << " int m=" << m << ", int n=" << n << ", m/n=" << m/n << ",
double(m)/n=" << double(m)/n;
myofs << " m/double(n)=" << m/double(n) << ",
double(m)/double(n)=" << double(m)/double(n) << endl << endl;
}
```

(конец листинга)

Результат работы кода, файл myproto.txt:

```
x02n05. Арифметические операции с явным преобразованием типов
int m=47, int n=8, m/n=5, double(m)/n=5.875 m/double(n)=5.875,
double(m)/double(n)=5.875
```

(конец вывода результата работы кода)

*Задание.* См. сборник задач.

## **1.5 Арифметические операторы С++.**

✓ *Страуструп С++*, §6.2.

✓ *Дейтел С++*, §1.18, §2.11, §2.12,

✓ *Либерти С++21*, глава 4.

### 1.5.1 Арифметические операции.

### 1.5.2 Операция копирования.

### 1.5.3 Операции сложения, умножения, вычитания, деления.

### 1.5.4 Операция вычисления остатка от деления нацело.

## **Практика 9. x02n06. Остаток от деления.**

Листинг кода (файл my.cpp)

```
{
myofs << "x02n06. Остаток от деления" << endl;
int m=47, n=8;
myofs << " int m=" << m << ", int n=" << n << ", m/n=" << m/n << ", m%n=" << m%n <<
endl << endl;
}
```

(конец листинга)

Результат работы кода, файл myproto.txt:

```
x02n06. Остаток от деления
int m=47, int n=8, m/n=5, m%n=7
```

(конец вывода результата работы кода)

**Задание.** См. сборник задач.

### 1.5.5 Операции инкремента ++ и декремента --.

#### **Практика 10. x02n11. Операции инкремента и декремента для int.**

**Листинг кода (файл my.cpp)**

```
{
  myofs << "x02n11. Операции инкремента и декремента для int" << endl;
  int m=23;
  myofs << " int m=" << m;
  int m1=m++;
  myofs << ", Если int m1=m++, то после этого m=" << m << ", m1=" << m1 << endl;
  int n=23;
  myofs << " int n=" << n;
  int n1=++n;
  myofs << ", Если int n1=++n, то после этого n=" << n << ", n1=" << n1 << endl;
  m=77;
  myofs << " int m=" << m;
  m1=m--;
  myofs << ", Если int m1=m--, то после этого m=" << m << ", m1=" << m1 << endl;
  n=77;
  myofs << " int n=" << n;
  n1=--n;
  myofs << ", Если int n1=--n, то после этого n=" << n << ", n1=" << n1 << endl;
  m=35;
  myofs << " int m=" << m;
  m1=m+=15;
  myofs << ", Если int m1=m++15, то после этого m=" << m << ", m1=" << m1 << endl;
  n=35;
  myofs << " int n=" << n;
  n1=n-=15;
  myofs << ", Если int n1=n-=15, то после этого n=" << n << ",
  n1=" << n1 << endl << endl;
}
```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

```
x02n11. Операции инкремента и декремента для int
int m=23, Если int m1=m++, то после этого m=24, m1=23
int n=23, Если int n1=++n, то после этого n=24, n1=24
int m=77, Если int m1=m--, то после этого m=76, m1=77
int n=77, Если int n1=--n, то после этого n=76, n1=76
int m=35, Если int m1=m++15, то после этого m=50, m1=50
int n=35, Если int n1=n-=15, то после этого n=20, n1=20
```

**(конец вывода результата работы кода)**

**Задание.** См. сборник задач.

#### **Практика 11. x02n11. Операции инкремента и декремента для double.**

**Листинг кода (файл my.cpp)**

```
{
  myofs << "x02n12. Операции инкремента и декремента для double" << endl;
  double x=23.4;
  double xpp = x+=3;
  myofs << " double x=" << x << ", double x1=x+=3, x1=" << xpp << ", x=" << x <<
  endl << endl;
}
```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

```
x02n12. Операции инкремента и декремента для double
double x=26.4, double x1=x+=3, x1=26.4, x=26.4
```

(конец вывода результата работы кода)

*Задание.* См. сборник задач.

### 1.5.6 Операции << и >>.

#### Практика 12. x02n07. Операция << для int.

Листинг кода (файл my.cpp)

```
{
  myofs << "x02n07. Операция << для int" << endl;
  int m=128, n=77;
  int m1=m<<1; int n1=n<<1;
  myofs << " int m=" << m << ", m<<1 =" << m1 << ", n=" << n << ", n<<1 =" << n1
  << endl;
  int m2=m<<2; int n2=n<<2;
  myofs << " int m=" << m << ", m<<2 =" << m2 << ", n=" << n << ", n<<2 =" << n2
  << endl << endl;
}
```

(конец листинга)

Результат работы кода, файл myproto.txt:

```
x02n07. Операция << для int
int m=128, m<<1 =256, n=77, n<<1 =154
int m=128, m<<2 =512, n=77, n<<2 =308
```

(конец вывода результата работы кода)

*Задание.* См. сборник задач.

#### Практика 13. x02n08. Операция >> для int.

Листинг кода (файл my.cpp)

```
{
  myofs << "x02n08. Операция >> для int" << endl;
  int m=128, n=77;
  int m1=m>>1; int n1=n>>1;
  myofs << " int m=" << m << ", m>>1 =" << m1 << ", n=" << n << ", n>>1 =" << n1
  << endl;
  int m2=m>>2; int n2=n>>2;
  myofs << " int m=" << m << ", m>>2 =" << m2 << ", n=" << n << ", n>>2 =" << n2
  << endl << endl;
  int m3=m>>3; int n3=n>>3;
  myofs << " int m=" << m << ", m>>3 =" << m3 << ", n=" << n << ", n>>3 =" << n3
  << endl << endl;
}
```

(конец листинга)

Результат работы кода, файл myproto.txt:

```
x02n08. Операция >> для int
int m=128, m>>1 =64, n=77, n>>1 =38
int m=128, m>>2 =32, n=77, n>>2 =19
int m=128, m>>3 =16, n=77, n>>3 =9
```

(конец вывода результата работы кода)

*Задание.* См. сборник задач.

### 1.5.7 Префиксные и постфиксные операторы.

### 1.5.8 Применение скобок.

### 1.5.9 Понятие функции. Основные действия, которые выполняет функция:

- Ж) Изменение значения полей данных объекта,
- К) Изменение порядка исполнения операторов (функций),
- Л) Изменение состояния внешних устройств.

### 1.5.10 Стандартные арифметические операторы. Функция sqrt(double).

#### Практика 14. x02n09. Функции sqrt(), floor(), ceil().

**Листинг кода (файл my.cpp)**

```
{
  myofs << "x02n09. Функции sqrt(), floor(), ceil()" << endl;
  double x=17.345;
  int m=floor(x);
  int n=ceil(x);
  myofs << " double x=" << x << ", sqrt(x)=" << sqrt(x) << ",
    floor(x)=" << m << ", ceil(x)=" << n << endl << endl;
}
```

**(конец листинга)****Результат работы кода, файл myproto.txt:**

```
x02n09. Функции sqrt(), floor(), ceil()
double x=17.345, sqrt(x)=4.16473, floor(x)=17, ceil(x)=18
```

**(конец вывода результата работы кода)****1.6 Преобразование арифметических типов.**

- ✓ *Страуструп С++, §B6.*
- ✓ *Дейтел С++, §1.18,*
- ✓ *Либерти С++21, глава 4.*

**1.6.1 Преобразование типов при выполнении арифметических операций.****1.6.2 Преобразование int в double.****Практика 15. x02n01. Операции с int и double константами.****Листинг кода (файл my.cpp)****(конец листинга)****Результат работы кода, файл myproto.txt:****(конец вывода результата работы кода)**

**Задание.** Разберитесь в значении фигурных скобок, обрамляющих код каждого упражнения.

**Задание.** Исправьте код так чтобы диагностики ошибок исчезли.

**Задание.** Создайте несколько переменных типа int, инициализируйте, выполните операции сложения, вычитания, умножения, деления, инкремента. Выведите результаты в файл. Создайте несколько переменных типа double, инициализируйте, выполните операции сложения, вычитания, умножения, деления, инкремента, декремента.

Выведите в файл протокола. Вычислите  $\frac{1 + 2 + 3 + 4 + 5}{4}$ ,

$\frac{1.0 + 2.0 + 3.0 + 4.0 + 5.0}{4.0}$ . Объясните разницу.

**1.6.3 Преобразование double в int. Функции double floor(double), double ceil(double).****Практика 16. x02n10. Функции sqrt(), floor(), ceil().****Листинг кода (файл my.cpp)**

```
{
  myofs << "x02n10. Функции sqrt(), floor(), ceil()" << endl;
  double x=16;
  int m=floor(x);
  int n=ceil(x);
  myofs << " double x=" << x << ", sqrt(x)=" << sqrt(x) << ",
    floor(x)=" << m << ", ceil(x)=" << n << endl << endl;
}
```

**(конец листинга)**

Результат работы кода, файл myproto.txt:

```
x02n10. Функции sqrt(), floor(), ceil()
double x=16, sqrt(x)=4, floor(x)=16, ceil(x)=16
```

(конец вывода результата работы кода)

### 1.6.4 Оператор typedef.

### 1.7 Вывод в файл значений арифметических типов.

✓ *Страуструн C++, §21.2.*

#### 1.7.1 Вывод int, double.

#### 1.7.2 Вывод с комментариями.

## Лекция 3. Операторы цикла.

### 1.8 Операторы цикла.

✓ *Страуструн C++, §6.3.*

✓ *Дейтел C++, §2.7– §2.15, §2.17– §2.18,*

✓ *Либерти C++21, глава 4, 7.*

#### 1.8.1 Цикл for. Итератор (переменная цикла).

**Практика 17. x03n01. Оператор цикла for с заданным числом итераций.**

Листинг кода (файл my.cpp)

```
{
myofs << "Упражнение 03-01. Оператор цикла for с заданным числом итераций." <<
endl;
int m, n=0;
double d=0;
for(m=1; m<124; m++)
{
d += 1.0/double(m);
n++;
}
myofs << "Число итераций n=" << n;
myofs << ", сумма " << n << " обратных натуральных чисел=" << d << endl;
myofs << "Значение переменной цикла m после его завершения m=" << m << endl <<
endl;
}
```

(конец листинга)

Результат работы кода, файл myproto.txt:

```
Упражнение 03-01. Оператор цикла for с заданным числом итераций.
Число итераций n=123, сумма 123 обратных натуральных чисел=5.393460
Значение переменной цикла m после его завершения m=124
```

(конец вывода результата работы кода)

**Задание.** Найдите сумму первых 100 натуральных чисел, 1...100.

#### 1.8.2 Инициализатор, условие, приращение.

**Практика 18. x03n02. Оператор цикла for с заданным приращением.**

Листинг кода (файл my.cpp)

```
{
myofs << "Упражнение 03-02. Оператор цикла for с заданным приращением." << endl;
int m, n=0;
double d=0;
for(m=1; m<124; m+=2)
{
d += 1.0/double(m);
n++;
}
```

```

}
myofs << "Число итераций n=" << n << ", сумма " << n << " обратных нечетных
натуральных чисел=" << d << endl;
myofs << "Значение переменной цикла m после его завершения m=" << m << endl <<
endl;
}

```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

Упражнение 03-02. Оператор цикла for с заданным приращением.

Число итераций n=62, сумма 62 обратных нечетных натуральных чисел=3.045328

Значение переменной цикла m после его завершения m=125

**(конец вывода результата работы кода)**

**Задание.** Найдите сумму нечетных натуральных чисел 1, 3, ..., 99.

### 1.8.3 О значении итератора после выхода из цикла for.

### 1.8.4 Применение счетчика числа итераций.

### 1.8.5 Оператор break.

#### **Практика 19. x03n03. Оператор цикла for и оператор break.**

**Листинг кода (файл my.cpp)**

```

{
myofs << "Упражнение 03-03. Оператор цикла for и оператор прерывания цикла break"
<< endl;
int n=0; double d=0;
for(int m=1; m<100000; m++)
{
d += 1.0/double(m);
n++;
if (d>10.0) break;
}
myofs << "Число итераций n=" << n << ", сумма " << n << " обратных натуральных
чисел=" << d << endl << endl;
}

```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

Упражнение 03-03. Оператор цикла for и оператор прерывания цикла break

Число итераций n=12367, сумма 12367 обратных натуральных чисел=10.000043

**(конец вывода результата работы кода)**

**Задание.** При каком наименьшем натуральном  $n$  значение суммы  $\sum_{k=1}^n \frac{1}{k}$  больше числа 100?

### 1.8.6 Оператор continue.

#### **Практика 20. x03n05. Оператор continue.**

**Листинг кода (файл my.cpp)**

```

{
myofs << "Упражнение 03-05. Оператор for и оператор continue" << endl;
int n=0; double d=0;
for(int m=1; m<1000; m++)
{
if (m%2==1) continue;
d += 1.0/double(m);
n++;
}
myofs << "Число итераций n=" << n << ", сумма " << n << " обратных нечетных
натуральных чисел=" << d << endl << endl;
}

```

**(конец листинга)**

Результат работы кода, файл myproto.txt:

Упражнение 03-05. Оператор for и оператор continue

Число итераций n=499, сумма 499 обратных нечетных натуральных чисел=3.395412

(конец вывода результата работы кода)

**Задание.** Найдите сумму натуральных чисел от 1 до 1000000, не делящихся нацело на 7.

### 1.8.7 Выход из цикла с помощью goto.

#### Практика 21. x03n04. Оператор goto.

Листинг кода (файл my.cpp)

```
{
myofs << "Упражнение 03-04. Прерывание оператора for с помощью оператора goto" <<
endl;
int n=0; double d=0;
for(int m=1; m<100000; m++)
{
d += 1.0/double(m);
n++;
if(d>10.0) goto next;
}
next:
myofs << "Число итераций n=" << n << ", сумма " << n << " обратных нечетных
натуральных чисел=" << d << endl << endl;
}
```

(конец листинга)

Результат работы кода, файл myproto.txt:

Упражнение 03-04. Прерывание оператора for с помощью оператора goto

Число итераций n=12367, сумма 12367 обратных нечетных натуральных чисел=10.000043

(конец вывода результата работы кода)

**Задание.** (задание с маленькой ловушкой) При каком наименьшем натуральном  $n$  значение

$$\text{суммы } \sum_{k=1}^n \frac{1}{k^2} \text{ больше числа } 100? .$$

### 1.8.8 О надежности С++ кода: определение итератора внутри оператора for.

### 1.8.9 Применение цикла для вывода в файл текстовой строки.

### 1.8.10 Применение цикла для вывода в файл набора чисел.

### 1.8.11 Циклы с несколькими инициализаторами.

#### Практика 22. x03n06. Оператор for с несколькими инициализаторами.

Листинг кода (файл my.cpp)

```
{
myofs << "Упражнение 03-06. Опасно!!! Оператор цикла for с несколькими
инициализаторами" << endl;
int m, n; double d;
for(m=1, n=0, d=0.0; m<1000; m++)
{
if (m%2==1) continue;
d += 1.0/double(m);
n++;
}
myofs << "Число итераций n=" << n << ", сумма " << n << " обратных нечетных
натуральных чисел=" << d << endl << endl;
}
```

(конец листинга)

Результат работы кода, файл myproto.txt:

Упражнение 03-06. Опасно!!! Оператор цикла for с несколькими инициализаторами

Число итераций  $n=499$ , сумма 499 обратных нечетных натуральных чисел=3.395412

(конец вывода результата работы кода)

**Задание.** Найдите одновременно сумму первых 100 натуральных чисел и их квадратов.

### **Практика 23. x03n07. Оператор for с несколькими инициализаторами.**

Листинг кода (файл my.cpp)

```
{
myofs << "Упражнение 03-07. Опасно!!! Оператор цикла for с несколькими
инициализаторами" << endl;
int n=7; double d=888;
for(int m=1, n=0, d=0.0; m<1000; m++){
    if (m%2==1) continue;
    d += 1.0/double(m);
    n++;
}
myofs << "Число итераций n=" << n << ", сумма " << n << " обратных нечетных
натуральных чисел=" << d << endl << endl;
}
```

(конец листинга)

Результат работы кода, файл myproto.txt:

Упражнение 03-07. Опасно!!! Оператор цикла for с несколькими инициализаторами  
Число итераций  $n=7$ , сумма 7 обратных нечетных натуральных чисел=888.000000

(конец вывода результата работы кода)

**Задание.** Найдите одновременно сумму первых 100 натуральных чисел и их квадратов.

### **1.8.12 Циклы с несколькими операторами приращения.**

### **1.8.13 Цикл while.**

### **Практика 24. x03n11. Оператор while.**

Листинг кода (файл my.cpp)

```
{
myofs << "Упражнение 03-11. Оператор while" << endl;
int n=0;
double d=0;
while(n<1000){
    n++;
    d+=1.0/double(n);
}
myofs << "Число итераций n=" << n << ", сумма " << n << " обратных натуральных
чисел=" << d << endl << endl;
}
```

(конец листинга)

Результат работы кода, файл myproto.txt:

Упражнение 03-11. Оператор while  
Число итераций  $n=1000$ , сумма 1000 обратных натуральных чисел=7.485471

(конец вывода результата работы кода)

**Задание.** .

### **1.8.14 Цикл while(true).**

### **Практика 25. x03n12. Оператор while(true).**

Листинг кода (файл my.cpp)

```
{
myofs << "Упражнение 03-12. Оператор while(true)" << endl;
int n=0; double d=0;
while(true)
{
    n++;
}
```

```

    d += 1.0/double(n);
    if(d>10.0)break;
}
myofs << "Число итераций n=" << n << ", сумма " << n << " обратных натуральных
чисел=" << d << endl << endl;
}

```

**(конец листинга)**

Результат работы кода, файл myproto.txt:

Упражнение 03-12. Оператор while(true)

Число итераций n=12367, сумма 12367 обратных натуральных чисел=10.000043

**(конец вывода результата работы кода)**

*Задание.*

### 1.8.15 Цикл do.

#### *Практика 26. x03n10. Оператор do-while.*

Листинг кода (файл my.cpp)

```

{
myofs << "Упражнение 03-10. Оператор do-while" << endl;
int n=0;
double d=0;
do{n++; d+=1.0/double(n);}
while(n<1000);
myofs << "Число итераций n=" << n << ", сумма " << n << " обратных натуральных
чисел=" << d << endl << endl;
}

```

**(конец листинга)**

Результат работы кода, файл myproto.txt:

Упражнение 03-10. Оператор do-while

Число итераций n=1000, сумма 1000 обратных натуральных чисел=7.485471

**(конец вывода результата работы кода)**

*Задание.*

### 1.8.16 Вложенные циклы.

#### *Практика 27. x03n08. Оператор for с несколькими итераторами.*

Листинг кода (файл my.cpp)

```

{
myofs << "Упражнение 03-08. Опасно!!! Оператор цикла for с несколькими итераторами"
<< endl;
int n=0; double d=0.0;
for(int m=1; m<1000; m++, n++, d+=1.0/double(m)){}
myofs << "Число итераций n=" << n << ", сумма " << n << " обратных натуральных
чисел=" << d << endl << endl;
}

```

**(конец листинга)**

Результат работы кода, файл myproto.txt:

Упражнение 03-08. Опасно!!! Оператор цикла for с несколькими итераторами

Число итераций n=999, сумма 999 обратных натуральных чисел=6.485471

**(конец вывода результата работы кода)**

*Задание.* Найдите одновременно сумму первых 100 натуральных чисел и их квадратов.

#### *Практика 28. x03n09. Оператор for с вещественным итератором.*

Листинг кода (файл my.cpp)

```

{
myofs << "Упражнение 03-09. Оператор цикла for с вещественным итератором" << endl;
double d=0.0;
int counter=0;

```

```
for(double x=1.5; x<10.0; x+=1.0, d+=x, counter++);
myofs << "Число итераций=" << counter << ", сумма=" << d << endl << endl;
}
```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

Упражнение 03-09. Оператор цикла for с вещественным итератором  
Число итераций=9, сумма=58.500000

**(конец вывода результата работы кода)**

**Задание.**

### **Практика 29. x03n13. Вложенные операторы цикла.**

**Листинг кода (файл my.cpp)**

```
{
myofs << "Упражнение 03-13. Вложенные операторы цикла." << endl;
const int M=12, N=5;
int m, n;
double a[M][N];
for(m=0; m<M; m++)
for(n=0; n<N; n++)
a[m][n] = m*n;
for(m=0; m<M; m++)
{
for(n=0; n<N; n++)
{
myofs << " a[" << setw(2) << m << ", " << setw(2) << n << "]=";
myofs << setw(5) << setprecision(1) << a[m][n];
}
myofs << endl;
}
}
}
```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

Упражнение 03-13. Вложенные операторы цикла.

```
a[ 0, 0]= 0.0 a[ 0, 1]= 0.0 a[ 0, 2]= 0.0 a[ 0, 3]= 0.0 a[ 0, 4]= 0.0
a[ 1, 0]= 0.0 a[ 1, 1]= 1.0 a[ 1, 2]= 2.0 a[ 1, 3]= 3.0 a[ 1, 4]= 4.0
a[ 2, 0]= 0.0 a[ 2, 1]= 2.0 a[ 2, 2]= 4.0 a[ 2, 3]= 6.0 a[ 2, 4]= 8.0
a[ 3, 0]= 0.0 a[ 3, 1]= 3.0 a[ 3, 2]= 6.0 a[ 3, 3]= 9.0 a[ 3, 4]= 12.0
a[ 4, 0]= 0.0 a[ 4, 1]= 4.0 a[ 4, 2]= 8.0 a[ 4, 3]= 12.0 a[ 4, 4]= 16.0
a[ 5, 0]= 0.0 a[ 5, 1]= 5.0 a[ 5, 2]= 10.0 a[ 5, 3]= 15.0 a[ 5, 4]= 20.0
a[ 6, 0]= 0.0 a[ 6, 1]= 6.0 a[ 6, 2]= 12.0 a[ 6, 3]= 18.0 a[ 6, 4]= 24.0
a[ 7, 0]= 0.0 a[ 7, 1]= 7.0 a[ 7, 2]= 14.0 a[ 7, 3]= 21.0 a[ 7, 4]= 28.0
a[ 8, 0]= 0.0 a[ 8, 1]= 8.0 a[ 8, 2]= 16.0 a[ 8, 3]= 24.0 a[ 8, 4]= 32.0
a[ 9, 0]= 0.0 a[ 9, 1]= 9.0 a[ 9, 2]= 18.0 a[ 9, 3]= 27.0 a[ 9, 4]= 36.0
a[10, 0]= 0.0 a[10, 1]= 10.0 a[10, 2]= 20.0 a[10, 3]= 30.0 a[10, 4]= 40.0
a[11, 0]= 0.0 a[11, 1]= 11.0 a[11, 2]= 22.0 a[11, 3]= 33.0 a[11, 4]= 44.0
```

**(конец вывода результата работы кода)**

**Задание.**

### **Практика 30. x03n14. Таблица умножения.**

**Листинг кода (файл my.cpp)**

```
{
myofs << "Упражнение 03-14. Таблица умножения." << endl;
const int M=12, N=5;
int m, n;
double a[M][N];
for(m=0; m<M; m++)
for(n=0; n<N; n++)
a[m][n] = m*n;
myofs << " ";
for(n=0; n<N; n++)
```

```

myofs << setw(6) << n;
myofs << endl;
for(m=0; m<M; m++)
{
myofs << setw(3) << m << " ";
for(n=0; n<N; n++)
myofs << setw(6) << setprecision(1) << a[m][n];
myofs << endl;
}
myofs << endl;
}

```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

Упражнение 03-14. Таблица умножения.

	0	1	2	3	4
0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	2.0	3.0	4.0
2	0.0	2.0	4.0	6.0	8.0
3	0.0	3.0	6.0	9.0	12.0
4	0.0	4.0	8.0	12.0	16.0
5	0.0	5.0	10.0	15.0	20.0
6	0.0	6.0	12.0	18.0	24.0
7	0.0	7.0	14.0	21.0	28.0
8	0.0	8.0	16.0	24.0	32.0
9	0.0	9.0	18.0	27.0	36.0
10	0.0	10.0	20.0	30.0	40.0
11	0.0	11.0	22.0	33.0	44.0

**(конец вывода результата работы кода)**

**Задание.** Составьте таблицы сложения и возведения в степень .

**Практика 31. х03п15. Анализ двумерной таблицы.**

**Листинг кода (файл my.cpp)**

```

{
myofs << "Упражнение 03-15. Анализ двумерной таблицы" << endl;
const int M=12, N=5;
double a[M][N];
double ar1[M], ar2[M], ac1[N], ac2[N];
int ir1[M], ir2[M], ic1[N], ic2[N];
int m, n, m1, m2, n1, n2;
double d1, d2, d3;
for(m=0; m<M; m++)
for(n=0; n<N; n++)
a[m][n] = 10.0*((double(rand())/double(RAND_MAX))*2.0-1.0);
for(m=0; m<M; m++)
{
d1=d2=a[m][0];
n1=n2=0;
for(n=1; n<N; n++)
{
d3=a[m][n];
if(d3<d1){d1=d3; n1=n;}
if(d3>d2){d2=d3; n2=n;}
}
ar1[m]=d1;
ar2[m]=d2;
ir1[m]=n1;
ir2[m]=n2;
}
for(n=0; n<N; n++)
{
d1=d2=a[0][n];
m1=m2=0;
}
}

```

```

    for(m=1; m<M; m++)
    {
        d3=a[m][n];
        if(d3<d1){d1=d3; m1=m;}
        if(d3>d2){d2=d3; m2=m;}
    }
    ac1[n]=d1;
    ac2[n]=d2;
    ic1[n]=m1;
    ic2[n]=m2;
}
myofs << " ";
for(n=0; n<N; n++){myofs << " [" << setw(2) << ic2[n] << "," << n << " "];} myofs <<
endl;
myofs << " ";
for(n=0; n<N; n++){myofs << setw(7) << setprecision(3) << ac2[n];} myofs << endl;
myofs << " -----" << endl;
for(m=0; m<M; m++){
    myofs << "a[" << setw(2) << m << "," << setw(1) << ir1[m] << "]= " << setw(7) <<
setprecision(3) << ar1[m] << " || ";
    for(n=0; n<N; n++){
        myofs << setw(7) << setprecision(3) << a[m][n];
    }
    myofs << " || a[" << setw(2) << m << "," << setw(1) << ir2[m] << "]= " << setw(7)
<< setprecision(3) << ar2[m] << endl;
}
myofs << " -----" << endl;
myofs << " ";
for(n=0; n<N; n++){myofs << " [" << setw(2) << ic1[n] << "," << n << " "];} myofs <<
endl;
myofs << " ";
for(n=0; n<N; n++){myofs << setw(7) << setprecision(3) << ac1[n];} myofs << endl;
}
myofs.close();
return 0;
}

```

**(конец листинга)**

**Результат работы кода, файл myproto.txt:**

```

Упражнение 03-15. Анализ двумерной таблицы
      [ 4,0] [ 2,1] [10,2] [10,3] [ 1,4]
      9.771  7.179  9.936  9.994  4.932
-----
a[ 0,0]= -9.975 || -9.975  1.272 -6.134  6.175  1.700 || a[ 0,3]=  6.175
a[ 1,1]= -2.994 || -0.403 -2.994  7.919  6.457  4.932 || a[ 1,2]=  7.919
a[ 2,0]= -6.518 || -6.518  7.179  4.210  0.271 -3.920 || a[ 2,1]=  7.179
a[ 3,0]= -9.700 || -9.700 -8.172 -2.711 -7.054 -6.682 || a[ 3,2]= -2.711
a[ 4,3]= -9.907 ||  9.771 -1.086 -7.618 -9.907 -9.822 || a[ 4,0]=  9.771
a[ 5,0]= -2.442 || -2.442  0.633  1.424  2.035  2.143 || a[ 5,4]=  2.143
a[ 6,4]= -8.859 || -6.675  3.261 -0.984 -2.958 -8.859 || a[ 6,1]=  3.261
a[ 7,4]= -3.961 ||  2.154  5.666  6.052  0.398 -3.961 || a[ 7,2]=  6.052
a[ 8,4]=  0.787 ||  7.519  4.534  9.118  8.514  0.787 || a[ 8,2]=  9.118
a[ 9,0]= -7.153 || -7.153 -0.758 -5.293  7.245 -5.808 || a[ 9,3]=  7.245
a[10,4]=  2.230 ||  5.593  6.873  9.936  9.994  2.230 || a[10,3]=  9.994
a[11,4]= -9.525 || -2.151 -4.676 -4.054  6.803 -9.525 || a[11,3]=  6.803
-----
      [ 0,0] [ 3,1] [ 4,2] [ 4,3] [ 4,4]
      -9.975 -8.172 -7.618 -9.907 -9.822

```

**(конец вывода результата работы кода)**

**Задание.**