

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| **Федеральное государственное бюджетное образовательное учреждение высшего**  **образования**  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА  Филиал РТУ МИРЭА в г. Ставрополе** |

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

к практическим занятиям и самостоятельной работе

по дисциплине «Практикум по разработке программного обеспечения»

для студентов направления «Информатика и вычислительная техника»

Направленность (профиль): «Программное обеспечение вычислительной техники и автоматизированных систем»

Квалификация (степень): бакалавр

Ставрополь 2018

Методические указания составлены в соответствии с требованиями государственного образовательного стандарта высшего образования и программой дисциплины «Практикум по разработке программного обеспечения» для студентов направления подготовки «Информатика и вычислительная техника»

Квалификация: бакалавр

Процесс изучения дисциплины направлен на формирование следующей компетенции:

|  |  |  |
| --- | --- | --- |
| **Направление**  **подготовки** | **Компетенции** | **Номер занятий, на которых реализуются компетенции** |
| 09.03.01  Информатика и вычислительная техника | ПК-2: способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования | Практические занятия 1-8 |

|  |  |
| --- | --- |
|  |  |

Составитель: Ю.В. Рокотов к.т.н., доцент

СОДЕРЖАНИЕ

Практическое занятие 1. Структура консольного приложения в C# 4

Практическое занятие 2. Предопределенные типы данных, переменные, константы 12

Практическое занятие 3. Использование возможностей консольного ввода-вывода. 21

Практическое занятие 4. Управление потоком выполнения с использованием операторов if, switch 27

Практическое занятие 5. Управление потоком выполнения с использованием оператора цикла for 33

Практическое занятие 6. Управление потоком выполнения с использованием операторов while 36

Практическое занятие 7. Управление потоком выполнения с использованием оператора do …while 41

Практическое занятие 8. Классы. Структуры. 44

Список рекомендуемой литературы 55

Практическое занятие 1. Структура консольного приложения в C#

1. Цель и содержание

Цель практического занятия: научиться работать с переменными и константами простых типов в C#.

Задачи практического занятия:

– научиться объявлять переменные простых типов в языке C#;

– научиться объявлять константы простых типов в языке C#;

– научиться выполнять простейшие действия с переменными и константами.

2. Формируемые компетенции

Практическое занятие направлено на формирование компетенции:

– способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования (ПК-2).

3. Теоретическая часть

Рассмотрим структуру консольного приложения на языке C#, созданного с использованием средств MS Visual Studio (рис. 1.1).

|  |
| --- |
| 4. Объявление метода Main  3. Объявление класса  2. Пользовательское пространство имен  1. Подключение пространств имен |
| Рисунок 1.1 – Исходный код консольного приложения. |

В интерактивном редакторе кода среды разработки код может быть свернут/ развернут с использованием кнопок +/-, внедренных в код. На рис. 1.1 в области 1 показано подключение пространств имен (библиотек, содержащих стандартные инструменты) с использованием зарезервированного слова using.

Программист сам создает свое пространство имен с именем LR\_One (2), в котором объявляется класс с именем Program.

В классе Program объявлен один метод – функция Main (параметры функции не рассматриваем).

Функция Main имеет особенное значение в программировании на языках C, C++ и C#.

Функцию Main называют «точкой входа», то есть началом выполнения программы. Далее мы рассмотрим приложения, содержащие множество функций. Операционная система знает с какой именно функции начать выполнение программы – с функции Main. Очевидно, что имя этой функции менять нельзя. Это должен быть статический метод класса (или структуры), возвращающий либо значение типа int, либо void. Хотя нередко модификатор public указывается явно, поскольку по определению этот метод должен быть вызван извне программы, на самом деле неважно, какой уровень доступа вы назначите методу точки входа. Он запустится, даже если вы пометите его как private.

Когда компилируется консольное или Windows-приложение С#, по умолчанию компилятор ищет в точности один метод Main () с описанной выше сигнатурой в любом классе и делает его точкой входа программы. Если существует более одного метода Main ( ), компилятор возвращает сообщение об ошибке.

Обратите внимание на вложенность конструкций на рис. 1.1: в пространство имен LR\_One вложен класс Program, в класс Program вложена функция Main. В свою очередь, в функции Main содержатся инструкции на языке C#-код, который начнет выполняться при старте программы.

В С#, как и в других С-подобных языках, большинство операторов завершаются точкой с запятой (;) и могут продолжаться в нескольких строках без необходимости указания знака переноса. Операторы могут быть объединены в блоки с помощью фигурных скобок ({ }). Однострочные комментарии начинаются с двойного слеша (//), а многострочные – сослеша со звездочкой (/\*) и заканчиваются противоположной комбинацией (\*/). В этих аспектах язык С# идентичен C++ и Java.

Следует также помнить о том, что язык С# чувствителен к регистру символов. Это значит, что переменные с именами myVar и MyVar являются разными.

Причина присутствия оператора using в файле Program.cs связана с использованием библиотечных классов.

Весь код С# должен содержаться внутри класса. Объявление класса состоит из ключевого слова class, за которым следует имя класса и пара фигурных скобок. Весь код, ассоциированный с этим классом, размещается между этими скобками.

4. Оборудование и материалы

Для выполнения практического занятия рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении практического занятия определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

1. Создайте консольное приложение, для этого выполните следующие действия:
   1. Выберите команду главного меню 
   2. В открывшемся диалоговом окне (рис. 1.2) выберите необходимые настройки для создаваемого проекта: язык Visual C#; фреймворк: .NET Framework 4; шаблон: Console Application.

|  |
| --- |
|  |
| Рисунок 1.2 – Создание нового проекта консольного приложения. |

* 1. В текстовом поле Name введите имя проекта (например LR\_One).
  2. В текстовом поле Location выберите место сохранения нового проекта.
  3. Установите флажок-переключатель «Create directory for solution».
  4. Нажмите кнопку «OK».

1. После выполнения пункта 1 в среде разработки откроется новый созданный проект (рис. 1.3).

|  |
| --- |
|  |
| Рисунок 1.3 – Новый проект, загруженный в среду разработки: вкладка «Solution Explorer» отображает состав проекта (нас интересует только файл Program.cs); в левой части окна (сверху) открыт файл Program.cs в редакторе кода. |

1. На данном этапе необходимо ознакомиться со структурой исходного файла консольного приложения (рис. 1.4).

|  |
| --- |
|  |
| Рисунок 1.4 – Исходный файл консольного приложения. |

1. Весь код программы необходимо писать внутри функции Main.
2. Для построения сборки (исполняемого exe-файла) выполните команду главного меню  (или использовать горячую клавишу ). После этого сборка создана, но приложение не будет запущено автоматически.
3. Для создания сборки и последующего запуска программы можно воспользоваться командой  главного меню среды разработки или нажать кнопку панели инструментов  . Можно также использовать горячую клавишу .
4. Запустите приложение на выполнение одним из методов, указанных в пункте 6. Окно консольного приложения появится и исчезнет. Это означает, что приложение выполнило все команды, написанные программистом, и завершило свою работу.
5. Для удержания окна на экране измените исходный файл в соответствии с рисунком 1.5. В функции Main добавлен вызов только одной команды Console.DeadKey ( ) – эта функция останавливает выполнение программы и ждет, когда пользователь нажмет любую клавишу.

|  |
| --- |
|  |
| Рисунок 1.5 – Исходный файл консольного приложения для предотвращения закрытия окна консольного приложения. |

1. Запустите измененное приложение, убедитесь, что окно удерживается на экране.
2. Добавьте несколько строк кода в исходный файл (рис. 1.6).

|  |
| --- |
|  |
| Рисунок 1.6 – Исходный файл консольного приложения для вывода информации на экран. |

1. Внимательно изучите исходный код примера на рис. 1.5. Запустите приложение и убедитесь, что отсутствуют ошибки и информация выводится.
2. Выполните задания к практическому занятию

Базовый уровень: Задания к практическому занятию 1:№ 1-7

Повышенный уровень: Задания к практическому занятию 1: № 8-11

**Задания к практическому занятию**

Измените приложение, созданное в ходе выполнения данной лабораторной работы таким образом, чтобы программа выводила на экран следующую информацию (каждый студент должен использовать персональную информацию о себе):

– Название и номер лабораторной работы;

– ФИО студента;

– Группа студента и шифр специальности;

– Дата рождения студента;

– Населенный пункт постоянного места жительства студента;

– Любимый предмет в школе;

– Краткое описание увлечений, хобби, интересов.

7. Содержание отчета и его форма

Отчет по практическому занятию должен содержать:

1. Номер и название практического занятия.

2. Цели практического занятия.

3. Ответы на вопросы для самостоятельной работы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения практического занятия, и результаты, полученные в ходе его выполнения.

Вопросы для самостоятельной работы

Базовый уровень:

1. Какая функция имеет особенное значение при выполнении программы на языках C, C++, C#?

2. Что такое «точка входа» в программе?

Повышенный уровень:

3. Как вы понимаете термины «пространства имен», «класс», «метод», «функция»?

4. Чем отличается структура консольного приложения от приложения, построенного с использованием технологий Windows Forms, WPF?

Практическое занятие 2. Предопределенные типы данных, переменные, константы

1. Цель и содержание

Цель практического занятия: научиться работать с переменными и константами простых типов в C#.

Задачи практического занятия:

– научиться объявлять переменные простых типов в языке C#;

– научиться объявлять константы простых типов в языке C#;

– научиться выполнять простейшие действия с переменными и константами.

2. Формируемые компетенции

Практическое занятие направлено на формирование компетенции:

– способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования (ПК-2).

3. Теоретическая часть

Перед выполнением лабораторной работы необходимо изучить материалы лекций. Следует понимать принцип деления типов .NET на типы значений и ссылочные типы.

На данном практическом занятии необходимо освоить приемы работы с предопределенными типами значений.

2.1 Типы значений C#

Язык C# поддерживает 8 предопределенных целочисленных типов (таблица 2.1).

Таблица 2.1 – Целочисленные типы C#.

|  |  |  |  |
| --- | --- | --- | --- |
| Имя типа | Тип CTS | Описание | Диапазон (минимум : максимум) |
| sbyte | System.SByte | 8-битное целое со знаком | -128 : 127 |
| short | System.Int16 | 16-битное целое со знаком | -32 768 : 32 767 |
| int | System.Int32 | 32-битное целое со знаком | -2 147 483 648 : 2 147 483 647 |
| long | System.Int64 | 64-битное целое со знаком | -263 : 263-1 |
| byte | System.Byte | 8-битное целое без знака | 0 : 255 |
| ushort | System.UInt16 | 16-битное целое без знака | 0 : 65 535 |
| uint | System.UInt32 | 32-битное целое без знака | 0 : 232-1 |
| ulong | System.UInt64 | 64-битное целое без знака | 0 : 264-1 |

Язык C# также поддерживает и типы с плавающей точкой (таблица 2.2).

Таблица 2.2 – Типы с плавающей точкой C#.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя типа | Тип CTS | Описание | Кол-во знаков | Диапазон (минимум : максимум) |
| float | System.Single | 32-битное с плавающей точкой одинарной точности | 7 | от  до |
| double | System.Double | 64-битное с плавающей точкой двойной точности | 15/16 | от  до |

В таблице 2.3 представлен десятичный тип C#. Данный тип реализован для финансовых операций.

Таблица 2.3 – Десятичный тип C#.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя типа | Тип CTS | Описание | Кол-во знаков | Диапазон (минимум : максимум) |
| decimal | System.Decimal | 128-битное с плавающей точкой в десятичной нотации с высокой точностью | 28 | от  до |

Как и во многих языках программирования существует булевский тип (таблица 2.4).

Таблица 2.4 – Булевский тип.

|  |  |  |
| --- | --- | --- |
| Имя типа | Тип CTS | Значения |
| bool | System.Boolean | true или false |

Для хранения одиночных символов в языке C# используется тип char (таблица 2.5)

Таблица 2.5 – Булевский тип.

|  |  |  |
| --- | --- | --- |
| Имя типа | Тип CTS | Значения |
| char | System.Char | Представляет отдельный 16-битный (Unicode) символ |

Литералы типа char записываются как одиночные, заключенные в одинарные кавычки символы: ‘F’, ‘w’, ‘ц’, ‘Я’ и т.д.

В переменных типа char можно хранить и специальные символы в виде управляющих последовательностей (таблица 2.6).

Таблица 2.6 – Представление символов в виде управляющих последовательностей.

|  |  |
| --- | --- |
| Управляющая последовательность | Символ |
| \’ | Одиночная кавычка |
| \” | Двойная кавычка |
| \\ | Обратный слэш |
| \0 | Пусто |
| \a | Предупреждение (звуковой сигнал) |
| \b | Забой |
| \f | Подача формы |
| \n | Новая строка |
| \r | Возврат каретки |
| \t | Символ табуляции |
| \v | Вертикальная табуляция |

Если отдельные символы объединены в строку, то необходимо использовать тип string, который отображается на тип CTS – System.String.

2.2 Объявление и инициализация переменных в C#

Синтаксис объявления переменных в C# выглядит следующим образом:



Например:



Этот код объявляет переменную типа int с именем a. Компилятор не позволит использовать эту переменную до тех пор, пока она не будет инициализирована (т.е. пока ей не будет присвоено значение).

Для инициализации переменной a необходимо написать следующий код:



Переменную можно инициализировать во время объявления:



или



Синтаксис C# позволяет объявить несколько переменных (и инициализировать их) одного типа в одной синтаксической конструкции.

Например:



В данном примере объявляется 5 переменных типа float, некоторые из них инициализируются в процессе объявления.

2.3 Объявление и инициализация констант в C#

Константа – это переменная, значение которой не меняется за время выполнения программы. Для объявления константы необходимо воспользоваться ключевым словом const. Например:



Очевидно, что при таком объявлении, поменять значения simv и pi в дальнейшем будет нельзя.

2.4. Оборудование и материалы

Для выполнения практического занятия рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

2.5. Указания по технике безопасности

Техника безопасности при выполнении практического занятия определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

2.6. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, описанном в практическом занятии №1.

2. Изучите материал в разделе «Теоретическая часть» данного практического занятия.

3. Модифицируйте исходный файл как показано на рис. 2.1.

|  |
| --- |
|  |
| Рисунок 2.1 – Объявление переменных в C#. |

4. Рассмотрим приведенный пример подробнее:

4.1. В строке 12 объявляется переменная типа int с именем a.

4.2. В строке 13 объявляется переменная типа int с именем b, причем при объявлении для нее устанавливается начальное значение, равное 7.

4.3. В строке 15 объявляется переменная типа string с именем str и инициализируется строковым значением «Hello, World!!!».

4.4. В строке 17 переменной a присваивается целочисленное значение 123.

4.5. Запустите приложение на выполнение. У вас должно появиться пустое окно консольного приложения. Очевидно, что исходный код, представленный на рис. 2.1 не предполагает вывода какой-либо информации на экран.

4.6. Для вывода информации на экран воспользуемся функцией Console.WriteLine, изученной на практическом занятии 1. Добавим в исходный файл следующие строки (строки 19 – 21 на рис. 2.2).

|  |
| --- |
|  |
| Рисунок 2.2 – Добавление строк кода для вывода значений объявленных переменных на экран. |

4.7. В примере на рис. 2.2 используется простой вывод, то есть имя переменной просто передается в качестве параметра функции Console.WriteLine.

4.8. Для выполнения форматного вывода (изменения формата представления выводимой информации) необходимо реализовать вывод в следующем виде (рис. 2.3 изменены строки 19-21):

|  |
| --- |
|  |
| Рисунок 2.3 – Вывод информации с использованием форматных строк. |

4.9. Внимательно изучите код полученной программы. Затем выполните задания к практическому занятию

**Задания к практическому занятию**

Объявите требуемые переменные, присвойте им начальные значения (определите самостоятельно, значения какого типа могут принимать переменные), выведите на экран с использованием форматной строки значения переменных и результат вычисления выражения в соответствие с вариантом:

|  |  |
| --- | --- |
| Вариант | Выражение для вычисления |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |

Базовый уровень: 1-12

Повышенный уровень: 13-25

7. Содержание отчета и его форма

Отчет по практическому занятию должен содержать:

1. Номер и название практического занятия.

2. Цели практического занятия.

3. Ответы на Вопросы для самостоятельной работы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения практического занятия, и результаты, полученные в ходе выполнения.

Вопросы для самостоятельной работы

Базовый уровень:

1. Что такое переменная? Как объявляется переменная?

2. Как объявляется константа? Чем константа отличается от переменной?

3. Какие типы значений применяются C#?

Повышенный уровень:

4. Чем тип char отличается от типа string?

5. Как производится инициализация переменных? Как производится инициализация констант?

6. Что такое управляющие последовательности?

Практическое занятие 3. Использование возможностей консольного ввода-вывода.

1. Цель и содержание

Цель практического занятия: изучить команды ввода и вывода данных в консольном приложении.

Задачи практического занятия:

– научиться применять функции Console.Write и Console.WriteLine;

– научиться применять функции Console.Read и Console.ReadLine;

– научиться использовать форматы вывода для различных типов данных.

2. Формируемые компетенции

Практическое занятие направлено на формирование компетенции:

– способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования (ПК-2).

3. Теоретическая часть

На предыдущих практических занятиях уже использовалась команда Console.WriteLine. Рассмотрим ее синтаксис подробнее:



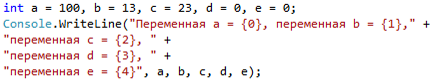
Эта запись означает вызов статического метода класса Console. В качестве параметра функции передается строка, которая выводится в консоль.

Существует также функция:



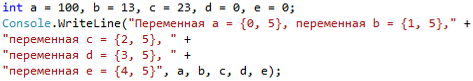
Этот метод также осуществляет вывод, только не переводит каретку на следующую строку.

Рассмотренные методы также позволяют осуществлять форматный вывод, аналогичный функции printf языка Си, например:



В данном примере в качестве первого параметра передается строка, содержащая маркеры в фигурных скобках. При выводе на места маркеров будут подставлены параметры, которые следуют за стрококй.

При форматном выводе можно задавать ширину поля вывода, для этого необходимо воспользоваться следующим приемом:



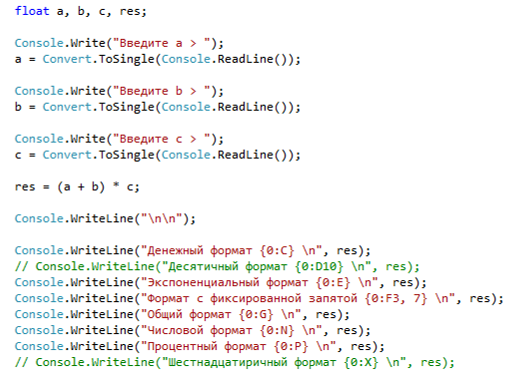
В данном примере видно, чтобы задать ширину вывода необходимо использовать в формате {n, w}, где n – порядковый номер параметра, а w – ширина области вывода (знак w позволяет осуществлять выравнивание выводимого значения по левому или правому краю).

При выводе также можно добавлять строку формата вместе с необязательным значением точности (таблица 2.1).

Таблица 3.1 – Основные строки формата.

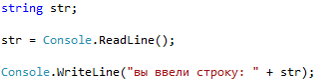
|  |  |
| --- | --- |
| Стока | Описание |
| C | Локальный формат валюты |
| D | Десятичный формат. Преобразует целое к основанию 10 и снабжает ведущими нулями, если есть спецификатор точности. |
| E | Научный (экспоненциальный) формат. Спецификатор точности устанавливает количество десятичных разрядов (по умолчанию 6). |
| F | Формат с фиксированной запятой. Спецификатор точности задает количество десятичных разрядов. |
| G | Общий формат. Форматирование E или F. |
| N | Числовой формат. Форматирует число с разделителями тысяч – запятыми. |
| P | Процентный формат |
| X | Шестнадцатиричный формат. Спецификатор точности используется для указания ведущих нулей. |

Пример использования формата:



Изучите представленный пример. Выполните данный код в среде выполнения.

В классе Console также существуют методы, позволяющие считывать информацию, вводимую пользователем:



В данном примере строка, введенная пользователем, считывается в переменную str, затее осуществляется ее вывод. Наберите данный пример в среде VS и изучите код.

4. Оборудование и материалы

Для выполнения практического занятия рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении практического занятия определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе №1.
2. Выполните индивидуальное задание, выполнив вывод результата с использованием всех возможных форматов, представленных в таблице 2.1.

**Задания к практическому занятию**

Объявите требуемые переменные, значения переменным пользователь должен присваивать в процессе выполнения программы (считываются с консоли), выведите на экран с использованием форматной строки значения переменных и результат вычисления выражения во всех возможных форматах:

|  |  |
| --- | --- |
| **Вариант** | **Выражение для вычисления** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Базовый уровень: 1-12

Повышенный уровень: 13-25

Вопросы для самостоятельной работы

Базовый уровень:

1. Как оформляется комментарий?
2. Какие функции используются для вывода информации в консоль?
3. Какие функции используются для считывания информации с консоли?

Повышенный уровень:

1. Какие форматы вывода вы знаете? Опишите их.
2. Какой класс содержит статические методы для конвертирования значений и приведения их к требуемому типу?
3. Какой класс содержит статические методы – математические функции? Что такое управляющие последовательности?

9. Список литературы

Для выполнения практического занятия, при подготовке к защите, а также для ответа на вопросы для самостоятельной работы рекомендуется использовать следующие источники: [4], [7], [8].

Практическое занятие 4. Управление потоком выполнения с использованием операторов if, switch

1. Цель и содержание

Цель практического занятия: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи практического занятия:

– научиться применять условныq оператор if;

– научиться применять оператор цикла for;

– научиться применять операторы выбора switch.

2. Формируемые компетенции

Практическое занятие направлено на формирование компетенции:

– способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования (ПК-2).

3. Теоретическая часть

**Условный оператор**. На предыдущих практических занятиях были рассмотрены вопросы программирования на языке C# с использованием только последовательного выполнения операторов программы.

В языке C# (как и в большинстве языков) можно вводить условия, циклы, ветвления.

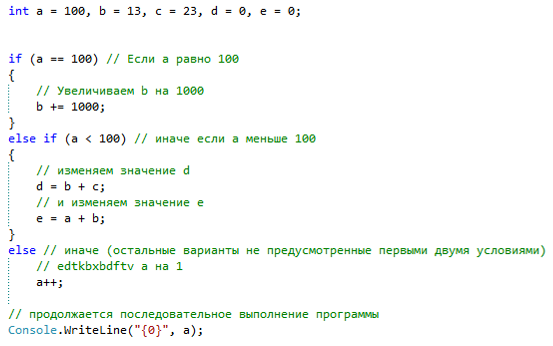
Условный оператор в С# позволяет организовать условие типа «если …, то …, иначе …».

Синтаксис условного оператора:



В данном синтаксисе  может быть составным оператором (группа операторов языка, заключенные в фигурные скобки). Условный оператор содержит только одно обязательное зарезервированное слово – if. Все остальные конструкции не являются обязательными (в синтаксическом описании на это указывают скобки [ ]).

Пример использования условного оператора:



В представленном примере сначала проверяется, является ли значение переменной a равным 100.

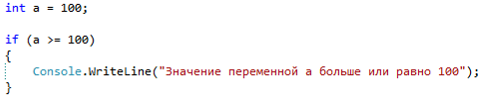
1. Если «Да», то выполняется единственный оператор (увеличение b на 1000). На этом выполнение всего условного оператора завершено и программа переходит к выполнению операторов, следующих за ним, то есть к выводу значения a.

2. Если первое условие не выполнилось, проверяется второе – является ли a меньше 100 (конструкция else if). Если это утверждение истинно, то выполняется составной оператор, состоящий из двух – изменение значений переменных d и e. Затем осуществляется переход к выводу переменной a.

3. Если не выполнилось ни одно условие с оператором if, то выполняется оператор, указанный после зарезервированного слова else. В данном случае это несоставной оператор (a увеличивается на 1), поэтому фигурные скобки можно не писать.

Во всей этой конструкции только оператор if является обязательным. Конструкций else if может быть любое количество, а if и else – только по одному.

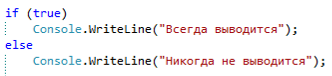
Еще один пример условного оператора:



В данном случае вывод в консоль выполнится, только если a больше либо равно 100.Иначе условный оператор ничего не выведет. Фигурные скобки в данном примере можно опустить.

Обратите внимание, что для проверки на равенство используется оператор ==, а не =. Знак «равно» используется в C# для присваивания значений. Следует понимать, что условное выражение, стоящее в конструкции if должно возвращать булево значение.

Например, следующий условный оператор всегда будет выполняться:

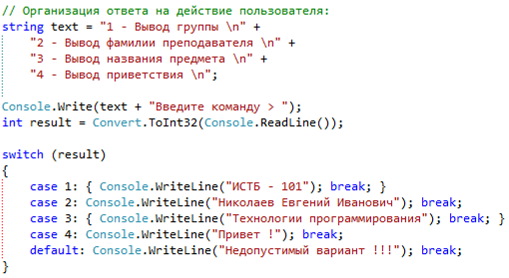


**Оператор выбора**. Синтаксис оператора:



В данном операторе осуществляется выбор оператора (который может быть составным) в зависимости от значения переменной . Если  равна значению , то выполнится , если  – , и т.д. Если ни одно значение, указанное в каком-либо операторе case, не совпало со значением , то выполнится оператор, указанный в секции . Внутри каждой секции case следует указать оператор break, который предотвращает проверку других условий после выполнения данного оператора. Следует обратить внимание, что в секциях case следует использовать только константы (переменные не допускаются).

Пример использование оператора switch … case:



Изучите представленный пример самостоятельно.

4. Оборудование и материалы

Для выполнения практического занятия рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении практического занятия определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в практическом занятии №1.
2. Выполните задания к практическому занятию. Во всех заданиях переменные X, Y являются вещественными и вводятся пользователем. Количество слагаемых также вводится пользователем. Программа должна вывести сумму заданного числа членов последовательности.

**Задания к практическому занятию**

Перед выполнением задания требуется самостоятельно определить закономерность изменения членов последовательности, чтобы применить цикл, условный оператор или, если потребуется, оператор выбора.

|  |  |
| --- | --- |
| Вариант | Выражение для вычисления |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 | (в знаменателе факториал) |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |

Базовый уровень: 1-12

Повышенный уровень: 13-25

Вопросы для самостоятельной работы

Базовый уровень:

1. Как оформляется комментарий?

2. Опишите синтаксис условного оператора. Какие части данного оператора являются обязательными?

Повышенный уровень:

3. Опишите синтаксис оператора выбора. Поясните почему следует использовать оператор break внутри каждого выбора.

4. Можно ли с помощью for реализовать бесконечный цикл? Поясните ответ на примерах.

Практическое занятие 5. Управление потоком выполнения с использованием оператора цикла for

1. Цель и содержание

Цель практического занятия: изучить операторы, позволяющие организовывать циклическое выполнение программного кода.

Задачи практического занятия:

– научиться применять оператор цикла for во вложенных циклах.

2. Формируемые компетенции

Практическое занятие направлено на формирование компетенции:

– способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования (ПК-2).

3. Теоретическая часть

Циклы позволяют выполнять определенную последовательность операторов до тех пор, пока выполняется некоторое условие. Каждый «круг» выполнения этого блока называется итерацией.

Синтаксис оператора:

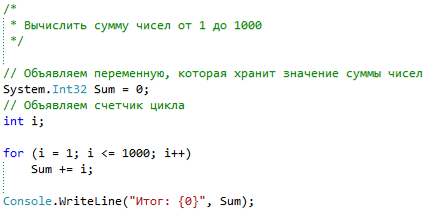


 – выражение, выполняемое перед первой итерацией цикла.

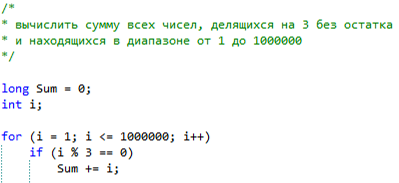
 – выражение булевого типа, которое проверяется перед каждой итерацией. Если оно истинно, то итерация выполняется, иначе – цикл завершается.

 – выражение, вычисляемое после каждой итерации.

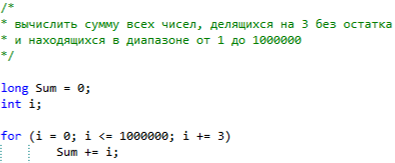
Пример применения оператора цикла for (пример 1):



Изучите код самостоятельно (создайте такую программу в VS). Еще один пример для самостоятельного изучения (пример 2):



Пример, отображающий возможность применения итератора (пример 3):



Обратите внимание, что примеры 2 и 3 решают одну и ту же задачу.

4. Оборудование и материалы

Для выполнения практического занятия рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении практического занятия определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

Необходимо модифицировать приложение, разработанное в рамках практического занятия №4 следующим образом:

1. Программа должна запросить исходные данные для вычисления выражения.

2. Программа вычисляет выражение и выводит результат.

3. Затем программа снова запрашивает ввод исходных данных и цикл повторяется.

С использованием цикла for организуйте указанное поведение программы и самостоятельно определите команду, которая приведет к завершению работы программы (например, нажата клавиша «Escape», пользователь должен ответить на вопрос «Продолжить?» и т.п.).

**Задания к практическому занятию**

В качестве индивидуального задания необходимо изменить код из практического занятия №4.

Вопросы для самостоятельной работы

Базовый уровень:

1. Как оформляется комментарий?

2. В разделе «Теоретическое обоснование» приведены примеры 1, 2 и 3. Почему в примерах 2 и 3 для переменной Sum выбран тип long, хотя в примере 1 для Sum выбран тип int? Обоснуйте ответ.

Повышенный уровень:

3. Опешите синтаксис оператора цикла for.

4. Можно ли с помощью for реализовать бесконечный цикл? Поясните ответ на примерах.

Практическое занятие 6. Управление потоком выполнения с использованием операторов while

1. Цель и содержание

Цель практического занятия: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи практического занятия:

– научиться применять оператор цикла с предусловием while;

– научиться применять оператор цикла с постусловием do … while.

2. Формируемые компетенции

Практическое занятие направлено на формирование компетенции:

– способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования (ПК-2).

3. Теоретическая часть

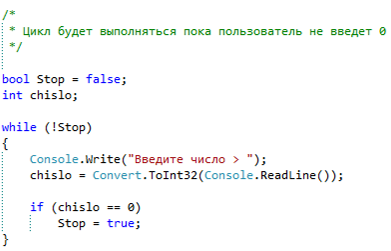
Цикл while похож на for тем, что является конструкцией с предварительной проверкой условия продолжения цикла. Но синтаксис цикла while более лаконичен:



В данном синтаксисе  может быть составным оператором (группа операторов языка, заключенные в фигурные скобки).

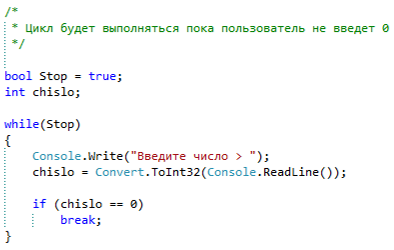
Следует понимать, что while используется для заранее неизвестного количества повторных выполнений операторов.

Пример выполнения цикла (пример 1):



Оператор break уже встречался в конструкции case оператора switch для выхода из case. Но break часто применяется внутри циклов for, while и do … while. Данный оператор прерывает выполнение цикла и передает управление оператору, следующему за циклом.

Пример, выполняющий то же самое, что и пример 1 (пример 2).



Следует обратить внимание, что цикл является бесконечным (логическая переменная Stop всегда является true), но выполнение цикла все равно прервется, если пользователь введет 0.

Оператор continue также применяется внутри цикла – он останавливает выполнение текущей итерации и немедленно переходит к выполнению следующей итерации.

4. Оборудование и материалы

Для выполнения практического занятия рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 2012 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении практического занятия определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в практическом занятии №1.
2. Выполните задания к практическому занятию. Во всех заданиях переменные X, Y являются вещественными и вводятся пользователем. Количество слагаемых пользователем не вводится. Программа должна работать следующим образом:

– пользователю выводится приглашение на ввод X и Y;

– пользователь вводит X и Y;

– программа начинает расчет суммы, при этом выводится результат расчета, полученный на каждой итерации;

– сначала выводится номер итерации (1) и сумма 1-го слагаемого, затем программа останавливается и ждет ввода команды пользователя (1 – продолжить, 0 – прекратить расчет);

– если пользователь продолжает, выводится номер итерации (2) и сумма 2-х слагаемых и снова программа ждет команды пользователя, и т.д.

1. Программа не должна использовать цикл for.

**Задания к практическому занятию**

Перед выполнением задания требуется самостоятельно определить закономерность изменения членов последовательности, чтобы применить цикл, условный оператор или, если потребуется, оператор выбора.

|  |  |
| --- | --- |
| Вариант | Выражение для вычисления |
| 1 | (в знаменателе факториал) |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |

Базовый уровень: 1-12

Повышенный уровень: 13-25

Вопросы для самостоятельной работы

1. Какой цикл лучше: while или for?

2. Опишите синтаксис оператора while.

3. Опишите синтаксис условного оператора. Какие части данного оператора являются обязательными?

4. Опишите синтаксис оператора выбора. Поясните, почему следует использовать оператор break внутри каждого выбора.

5. Поясните назначение операторов break и continue.

Практическое занятие 7. Управление потоком выполнения с использованием оператора do …while

1. Цель и содержание

Цель практического занятия: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи практического занятия:

– научиться применять оператор цикла с постусловием do … while.

– изучить особенности оператора do…while;

– научиться использовать оператор цикла do…while для решения практических задач.

2. Формируемые компетенции

Практическое занятие направлено на формирование компетенции:

– способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования (ПК-2).

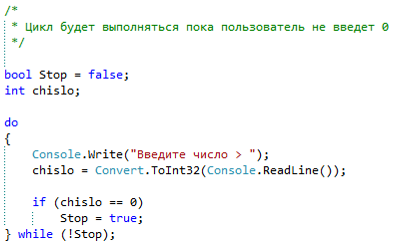
3. Теоретическая часть

Цикл do … while полностью повторяет функциональные возможности while, но предполагает проверку условия окончания цикла после выполнения тела цикла. То есть блок операторов тела цикла всегда выполнится хотя бы один раз.

Синтаксис оператора:



Пример использования цикла (пример 1):



4. Оборудование и материалы

Для выполнения практического занятия рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении практического занятия определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в практическом занятии №1.
2. Выполните задания к практическому занятию. Во всех заданиях переменные X, Y являются вещественными и вводятся пользователем. Количество слагаемых пользователем не вводится. Программа должна работать следующим образом:

– пользователю выводится приглашение на ввод X и Y;

– пользователь вводит X и Y;

– программа начинает расчет суммы, при этом выводится результат расчета, полученный на каждой итерации;

– сначала выводится номер итерации (1) и сумма 1-го слагаемого, затем программа останавливается и ждет ввода команды пользователя (1 – продолжить, 0 – прекратить расчет);

– если пользователь продолжает, выводится номер итерации (2) и сумма 2-х слагаемых и снова программа ждет команды пользователя, и т.д.

1. Программа не должна использовать цикл for или while.

**Задания к практическому занятию**

Необходимо использовать индивидуальное задание из практического занятия №6.

Вопросы для самостоятельной работы

Базовый уровень:

1. Какой цикл лучше do…while или while?

2. Опишите синтаксис оператора do…while.

3. Опишите синтаксис условного оператора. Какие части данного оператора являются обязательными?

Повышенный уровень:

4. Опишите синтаксис оператора выбора. Поясните, почему следует использовать оператор break внутри каждого выбора.

5. Опешите синтаксис оператора do … while.

6. Поясните назначение операторов break и continue.

\

Практическое занятие 8. Классы. Структуры.

1. Цель и содержание

Цель практического занятия: изучить структуру и принципы объявления классов, освоить технологию создания экземпляров классов (объектов).

Задачи практического занятия:

– научиться объявлять классы;

– научиться создавать объекты классов;

– научиться работать с полями данных и методами классов.

2. Формируемые компетенции

Практическое занятие направлено на формирование компетенции:

– способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования (ПК-2).

3. Теоретическая часть

3.1 Классы и структуры.

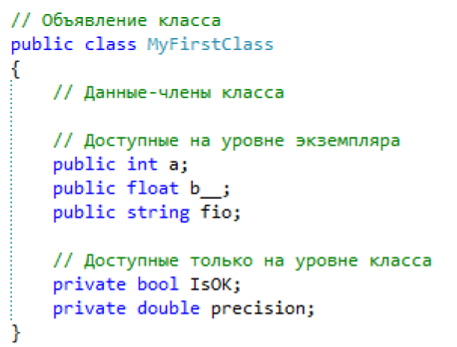
Класс – это тип данных, объединяющий данные и методы их обработки. Класс – это пользовательский шаблон, в соответствии с которым можно создавать объекты. То есть класс – это правило, по которому будет строиться объект. Сам класс не содержит данных.

Объект класса (экземпляр класса) – переменная типа класс. Объект содержит данные и методы, манипулирующие этими данными. Класс определяет, какие данные содержит объект и каким образом он ими манипулирует.

Все, что справедливо для классов, можно распространить и на структуры. Отличие состоит в методе хранения объектов данных типов в оперативной памяти: структуры – это типы по значению, они размещаются в стеке; классы – это ссылочные типы, объекты классов размещаются в куче. Структуры не поддерживают наследование.

Структуры применяются для представления небольших объемов данных. Объявление структур происходит с использованием ключевого слова struct, объявление классов – с помощью ключевого слова class.

Пример объявления класса:



При создании как классов, так и структур, используется ключевое слово new, например:



3.2 Структура класса

Данные и функции, объявленные внутри класса, называются членами класса (class members). Доступность членов класса может быть описана как public, private, protected, internal или internal protected.

3.2.1 Данные-члены.

Данные-члены – это те структуры внутри класса, которые содержат данные класса – поля, константы события.

Поля – это любые переменные, ассоциированные с классом. После создания экземпляра класса к полям можно обращаться с использованием синтаксиса , например: ИмяПоля ИмяОбъекта.



Аналогичным образом с классом ассоциируются константы.

События будут рассмотрены в следующих лабораторных работах.

3.2.2 Функции-члены

Функции-члены – это члены, которые обеспечивают некоторую функциональность для манипулирования данными классов. Они делятся на следующие виды: методы, свойства, конструкторы, финализаторы, операции и индексаторы.

Методы (method) – это функции, ассоциированные с определенным классом. Как и данные-члены, по умолчанию они являются членами экземпляра. Они могут быть объявлены статическими с помощью модификатора static.

Свойства (property) – это наборы функций, которые могут быть доступны клиенту таким же способом, как общедоступные поля класса. В С# предусмотрен специальный синтаксис для реализации чтения и записи свойств для классов, поэтому писать собственные методы с именами, начинающимися на Set и Get, не понадобится. Поскольку не существует какого-то отдельного синтаксиса для свойств, который отличал бы их от нормальных функций, создается иллюзия объектов как реальных сущностей, предоставляемых клиентскому коду.

Конструкторы (constructor) – это специальные функции, вызываемые автоматически при инициализации объекта. Их имена совпадают с именами классов, которым они принадлежат, и они не имеют типа возврата. Конструкторы полезны для инициализации полей класса.

Финализаторы (finalizer) похожи на конструкторы, но вызываются, когда среда CLR определяет, что объект больше не нужен. Они имеют то же имя, что и класс, но с предшествующим символом тильды (~). Предсказать точно, когда будет вызван финализатор, невозможно.

Операции (operator) – это простейшие действия вроде + или -. Когда вы складываете два целых числа, то, строго говоря, применяете операцию + к целым. Однако С# позволяет указать, как существующие операции будут работать с пользовательскими классами (так называемая перегрузка операций).

Индексаторы (indexer) позволяют индексировать объекты таким же способом, как массив или коллекцию.

На данном практическом занятии рассматриваются только методы класса – это функции, ассоциированные с определенным классом.

В C# объявление метода класса состоит из спецификатора доступности, возвращаемого значения, имени метода, списка формальных параметров и тела метода:



Например, добавим методы для объявленного ранее класса MyFirstClass:



Синтаксис вызова методов аналогичен синтаксису обращения к данным-членам:



В данном примере метод InitClassMembers не возвращает никаких данных, но требует передачи ему фактических параметров. В свою очередь, метод GetAbsA возвращает значение типа int и не предполагает никаких параметров.

В общем случае параметры могут передаваться методу либо по значению, либо по ссылке. Когда переменная передается по ссылке, вызываемый метод получает саму переменную, поэтому любые изменения, которым она подвергнется внутри метода, останутся в силе после его завершения. Но если переменная передается по значению, вызываемый метод получает копию этой переменной, а это значит, что все изменения в ней по завершении метода будут утеряны. Для сложных типов данных передача по ссылке более эффективна из-за большого объема данных, который приходится копировать при передаче по значению.

Если не указано обратное, то в С# все параметры передаются по значению. Тем не менее, можно принудительно передавать значения по ссылке, для чего используется ключевое слово ref. Если параметр передается в метод, и входной аргумент этого метода снабжен префиксом ref, то любые изменения этой переменной, которые сделает метод, отразятся на исходном объекте.

В С-подобных языках функции часто возвращают более одного значения. Это обеспечивается применением выходных параметров, за счет присваивания значений переменным, переданным в метод по ссылке. Часто первоначальное значение таких переменных не важно. Эти значения перезаписываются в функции, которая может даже не обращать внимания на то, что в них хранилось первоначально.

Было бы удобно использовать то же соглашение в С#. Однако в С# требуется, чтобы переменные были инициализированы каким-то начальным значением перед тем, как к ним будет выполнено обращение. Хотя можно инициализировать входные переменные какими-то бессмысленными значениями до передачи их в функцию, которая наполнит их осмысленными значениями, этот прием выглядит в лучшем случае излишним, а в худшем – сбивающим с толку. Тем не менее, существует способ обойти требование компилятора С# относительно начальной инициализации переменных.

Это достигается ключевым словом out. Когда входной аргумент снабжен префиксом out, этому методу можно передать неинициализированную переменную. Переменная передается по ссылке, поэтому любые изменения, выполненные методом в переменной, сохраняются после того, как он вернет управление. Ключевое слово out также должно указываться при вызове метода – так же, как при его определении.

**Частичные классы**. Ключевое слово partial (частичный) позволяет определить класс, структуру или интерфейс, распределенный по нескольким файлам. Но ситуации, когда множеству разработчиков требуется доступ к одному и тому же классу, или же в ситуации, когда некоторый генератор кода генерирует часть класса, такое разделение класса на несколько файлов может оказаться полезным. Ключевое слово partial просто помещается перед классом, структурой или интерфейсом.

**Статические классы**. Статический класс функционально представляет собой то же самое, что и класс с приватным статическим конструктором. Создать экземпляр такого класса невозможно. Если указать ключевое слово static в объявлении класса, компилятор будет гарантировать, что к этому классу никогда не будут добавлены нестатические члены.

**Класс Object**. Классы .NET изначально унаследованы от System.Object. Фактически, если при определении нового класса базовый класс не указан, компилятор автоматически предполагает, что он наследуется от Object.

Практическое значение: помимо методов и свойств, определяемых программистом самостоятельно, также появляется доступ к множеству общедоступных и защищенных методов-членов, которые определены в классе Object. Эти методы присутствуют во всех определяемых классах.

4. Оборудование и материалы

Для выполнения практического занятия рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении практического занятия определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения занятия

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в практическом занятии №1.

2. Изучите пример выполнения задания, представленный в данном разделе.

3. Выполните задания к практическому занятию. Задания ориентированы на работу с классами.

**Пример выполнения задания.**

Разработать класс для представления объекта «Прямоугольный параллелепипед». Реализуйте все необходимые поля данных (закрытые) и методы, позволяющие:

– устанавливать и считывать значения полей данных;

– вычислять объем прямоугольного параллелепипеда;

– вычислять площадь поверхности прямоугольного параллелепипеда;

– выводить полную информацию об объекте в консоль.

Решение данной задачи состоит из двух этапов: объявление класса Parallelepiped и демонстрация использования объекта данного класса.

Полный листинг примера:







В данном примере необходимо обратит внимание на тот факт, что все вычисления выполняются внутри класса. Метод Main содержит только вызовы методов класса, то есть вся реализация скрыта.

В результате выполнения программы отобразится следующее консольное окно с выводом информации:



**Задания к практическому занятию**

Спроектируйте класс, наполните его требуемой функциональностью, продемонстрируйте работоспособность класса.

|  |  |
| --- | --- |
| Вариант | Выражение для вычисления |
| **1.** | Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| **2.** | Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте. |
| **3.** | Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| **4.** | Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте. |
| **5.** | Класс «График y=x». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте. |
| **6.** | Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| 7. | Класс «Матрица M×N». Реализовать инициализацию элементов матрицы случайными числами, вывод матрицы, нахождение максимального и минимального элементов, а также вывод информации об объекте. |
| **8.** | Класс «Прямоугольный треугольник». Реализовать ввод и вывод полей данных, вычисление гипотенузы, площади и периметра, а также вывод информации об объекте. |
| **9.** | Класс «Отрезок». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте. |
| **10.** | Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте. |
| **11.** | Класс «Ромб». Реализовать ввод и вывод полей данных (диагонали ромба), вычисление площади, периметра, а также вывод информации об объекте. |
| **12.** | Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте. |
| **13.** | Класс «График y=3x+5». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте. |
| **14.** | Класс «Матрица M×N». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение среднего арифметического всех элементов, а также вывод информации об объекте. |
| **15.** | Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте. |
| **16.** | Класс «График y=x-10». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте. |
| **17.** | Класс «Матрица M×N». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение и вывод среднего арифметического элементов в каждом столбце. |
| **18.** | Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте. |
| **19.** | Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| **20.** | Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте. |
| **21.** | Класс «График y=x». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте. |
| **22.** | Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте. |
| **23.** | Класс «График y=-x». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте. |
| **24.** | Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте. |
| **25.** | Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте. |

Базовый уровень: 1-12

Повышенный уровень: 13-25

Вопросы для самостоятельной работы

Базовый уровень:

1. Что такое класс?

2. Что такое структура? Чем структура отличается от класса?

3. Что такое члены класса? Какие группы членов класса вы знаете?

4. Какие типы членов-данных вы знаете?

Повышенный уровень:

5. Какие типы функций-членов класса вы знаете?

6. Как поменять цвет текста в консольном приложении?

7. Как поменять цвет фона в консольном приложении?

8. Какие модификаторы доступности членов класса вы знаете?

9. Какое ключевое слово используется для создания объекта класса?

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Основная:

1. Библия C#: Пособие / Фленов М.Е., - 3-е изд., перераб. и доп. - СПб: БХВ-Петербург, 2016. - 544 с. - Режим доступа: http://znanium.com/catalog/product/944563

2. Программирование на С++/С# в Visual Studio .NET 2003: Пособие / Понамарев В.А. - СПб: БХВ-Петербург, 2015. - 340 с. - Режим доступа: http://znanium.com/catalog/product/939605

Дополнительная:

3. Введение в программирование на языке Visual C# : учеб. пособие / С.Р. Гуриков. – М.: ФОРУМ: ИНФРА-М, 2017. – 447 с. - Режим доступа: http://znanium.com/catalog/product/752394

4. Самоучитель С#.: Пособие / Секунов Н.Ю. - СПб: БХВ-Петербург, 2014. - 564 с. - Режим доступа: http://znanium.com/catalog/product/939986