О курсе

Чему учит курс?

После курса, вы сможете решать задачи по программированию на ЕГЭ, выиграть несложную олимпиаду по информатике, а также начать свой профессиональный путь в качестве программиста.

Язык программирования

В курсе мы будем изучать язык программирования C#, так как этот язык очень прост и хорошо подходит для новичков. На сегодняшний момент C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. На нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

Основные понятия

Перед тем, как переходить к программированию, необходимо разобрать основные понятия и приемы, которые пригодятся в программировании:

**Алгоритм** - это последовательность действий для достижения поставленной цели.

**Язык** **программирования** - это язык, который понимает компьютер. Есть русский, английский и другие языки которые понимаем мы, а есть другие языки - языки программирования, которые понимает компьютер.

**Программа** - алгоритм, записанный на некотором языке программирования.

1. этапы решения задачи

Решение любой математической задачи с помощью программирования можно свести к следующим этапам:

1. **Постановка** **задачи** - описание самой задачи
2. **Формализация** - перевод на математический язык
3. **Алгоритмизация** - придумывание алгоритма, который решает задачу
4. **Программирование** - написание кода программы
5. **Тестирование** - тестирование программы.

Советуется всегда решать задачи по данным этапам. Так вы сэкономите массу времени и напишите правильные программы без ошибок.

Идеология программ

Полезные программы работают следующим образом:

* **Считывают** информацию. В нашем случае программа будет считывать информацию с консоли. Данная информация называется - входные данные, то есть, что получает программа на вход;
* Программа **обрабатывает** входные данные по заложенному алгоритму;
* **Выводит** информацию. В нашем случае выводить программа будет тоже на консоль. Данные, которые выводит программа называются выходными данными.

Среда разработки

Установка Visual Studio

Для того, чтобы начать создавать программы на C#, вам необходима среда разработки – то где вы будете непосредственно писать код, компилировать его и отлаживать. В этом курсе мы будем писать программы в **Microsoft Visual Studio 2017 Community Edition**. Это бесплатный инструмент для некоммерческого использования. Вы можете пользоваться любой версией, так как отличия версий невелики.

Создание проекта

Теперь у вас есть среда разработки. Давайте создадим проект, в котором вы будете писать программы. Давайте сделаем это вместе в среде разработки.

Чтобы создать проект, нажмите «**Файл**», а затем «**Создать проект**».

В появившемся окне слева выберите платформу Visual C#, тип приложения «**Консольное приложение**», затем назовите проект на английском языке в поле «**Имя решения**» в самом низу открывшегося окна и жмем "**ОК**".

Первая программа

Сгенерированный шаблон

Visual Studio сгенерировала следующий шаблон кода:

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace FirstApp  
{

class Program

{

static void Main(string[] args)

{

}

}  
}

﻿И это уже готовая программа, только она не делает ничего.

Посмотрим, из чего состоит программа. В первых четырех строках записаны команды **using**. Команда **using** означает, что мы хотим обратиться к библиотеке, содержащей разные функции и разные полезные вещи, которые написали за нас другие программисты. Нам понадобится толькопервая библиотека "**System**", с помощью которой наша программа сможет общаться с внешним миром. Остальные три при желании можно стереть.

Перейдем сразу к тому, что нам точно понадобится.

static void Main(string[] args)//Заголовок(название) функции

{ // Тело функции

} //

**Main** - это основная функция программы, она будет запускаться автоматически при запуске программы. В программае может быть несколько функций , но первой всегда будет запускаться **Main**. Пока же это основная и единственная функция. Внутри фигурных скобок (тела функции) мы будем писать свои программы, то есть начинаться после открывающей и заканчиваться до закрывающей.

Структуру языка **C#** можно разделить на две части – **структуры** и **команды** и оформляются они следующим образом:

* После структуры ставятся парные фигурные скобки **({})**, между которыми находится их тело;
* После каждой команды нужно ставить точку с запятой **(;)**.

Так же в C#, как и в большинстве языков программирования есть **комментарии**. **Комментарии** позволяют отделять код программы и любой другой текст.

Все, что находится в **однострочном** **комментарии** — от **//** до конца строки — игнорируется компилятором, как и весь **многострочный** **комментарий**, расположенный между **/\*** и **\*/**. Очевидно, что в многострочном комментарии не может присутствовать комбинация **\*/**, поскольку она используется как конец комментария.

Hello, World!

Программа, которая не делает ничего, у нас уже есть. В программировании есть традиция – изучение любого языка начинается с написания программы, которая выводит на экран сообщение «**Hello, World!**».

Мы не будем отступать от традиций:

using System;  
  
namespace FirstApp  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            Console.Write("Hello, World!");  
        }  
    }  
}

У нас появилась новая строка:

Console.Write("Hello, World!");

Она выводит на экран текст «**Hello, World!**». Разберемся в программе подробнее.

**Console** – это объект, который отвечает за консоль. **Консоль** - это черное окошко, которое появляется при запуске программы. Операция "**Write**" как раз и означает «**написать**». Не случайно программисты так назвали, ведь мы хотим вывести (написать) в консоль, а с английского «**Console Write**» переводится «**Написать на консоль**». Далее идет текст внутри круглых скобок и в кавычках. Не забывайте ставить точку с запятой (**;**) там, где она есть в примерах кода.

**Строка выведется ровно такая, как указали внутри кавычек, со всеми пробелами и символами.**

Запуск программы

Давайте запустим нашу программу. Для этого необходимо нажать комбинацию клавиш **CTRL** и **F5**. Если в программе есть какая-либо ошибка, система сообщит нам, что запуск не удался. Ошибки будут перечислены в окне «**Список ошибок**». Если же всё правильно, то появится чёрное окно консоли с надписью "**Hello, World!**".

Вывод строк

Write и WriteLine

Кроме **Console.Write()** есть команда **Console.WriteLine()**, которая выводит точно также переданную строку, но кроме этого, еще сдвигает текущий курсор в консоли на новую строку.

Для лучшего понимания рассмотрим два примера:

static void Main(string[] args)  
{  
   Console.Write("Hello");

  Console.Write("World!");

}

static void Main(string[] args)  
{

   Console.WriteLine("Hello");

  Console.WriteLine("World!");  
}

Первый пример использует уже известный нам **Console.Write**. Данная программа выведет на одной строке склеенныестроки, то есть:

HelloWorld!

Дело в том, что после первой команды **Console.Write()** курсор в консоли остается сразу после слова **Hello**, а вторая команда при выводе дописывается свою строку. Таким образом получается склеенная строка.

А вторая программа используется **Console.WriteLine()**. Данная команда после вывода переводит курсор на следующую строку, то есть, вывод будет следующим:

Hello

World!

Вывод данных

Точно так же можно выводить числаили результат вычисления выражений. Например:

static void Main(string[] args)

{  
  Console.WriteLine(26);   
  Console.WriteLine(1024 + 48);  
  Console.WriteLine(5 + 8 \* 2);  
  Console.Write((5 + 8) \* 2);

}

Как вы могли заметить, каждый результат выводится на каждой строке. Из представленной программы видно, что операциинад числами делаются точно также как в математике.

Форматированный вывод

Довольно часто надо бывает выводить числа и строки вместе. Для этого между числом и строкой нужно поставить знак плюс. Для лучшего понимания, рассмотрим код:

static void Main(string[] args)

{   
  Console.WriteLine("5 + 7 = " + 12);

  Console.WriteLine(5 + " + " + 7 + " = " + 12);   
  Console.WriteLine(3 + " " + 6);

}

Заметьте, что пробел (**" "**) это тоже строка, причем **не пустая**. Пример показывает, что один и тот же результат вывода можно достичь разными способами.

Переменные

Программа, которая всегда считает результат вычисления одного и того же выражения, довольно скучная и бессмысленная. Полезная программа должна оперировать с различными данными без внесения изменений в код.

Для хранения информации в программировании используются **переменные**. **Переменную** можно рассматривать как **ящик**. Однажды сделав такой ящик мы можем класть в него разные вещи. Под каждый тип информации - нужен ящик соответствующего типа, вы ведь не будете складывать деньги, спички, бензин и шоколад в один и тот же ящик. Таким образом, у каждой переменной есть **тип**, который надо указать при ее создании. Также у переменной есть **имя**, по которому мы будем к нему обращаться.

То есть, **переменная** - это ячейка памяти определенного **типа**, имеющая **имя**.

Типы данных

Кроме целого числа, существуют множество типов данных, однако мы рассмотрим основные типы данных:

* **int** - целое число;
* **string** – строка;
* **double** - число с дробной частью;
* **char** – символ;
* **bool** - специальный тип, принимает только два значения - **true** или **false.**

Чтобы создать переменную, которая будет хранить **целые числа**, нужно написать:

int a;

Данной строкой кода, мы **объявили переменную**, указали, что там будут хранятся **целые числа** и обозвали его "**a**".

Теперь в созданную переменную можем записывать только **числа.** Чтобы записать в переменную целое число, нужно воспользоваться следующим правилом **куда = что,** где "**=**" - **оператор присвоения.** Воспользуемся данным правилом. Например, запишем в переменную число 7:

a = 7;

Операции производятся справа налево- взять число 7 и записать в переменную. Эти два шага можно объединить, то есть можно сразу объявить и проинициализировать, то есть присвоить начальное значение:

int a = 7;

Мы объявили переменную **a** и записали сразу значение 7.   Чтобы узнать **содержимое переменной**, нужно обратиться к нему по **имени**. Например:

int b = a \* 5;

В переменную запишется значение 35, так как вместо переменной **a** подставится его **значение**, то есть **7**. Также мы можем поменять значение переменной a:

a = a + 8;

Действия выполняются справа налево, значит мы берем значение переменной **а** которое равно 7, к нему добавляем 8 и снова записываем в переменную **а**. Таким образом, значение переменной увеличили на 8.

Несколько переменных **одного типа** можно записывать через запятую:

int a, b = 1, c = 2, d;

Давайте составим программу, для лучшего понимания материала:

int a = 6;  
int b = a \* 8;  
a = b - 8;   
int c = b + a;   
Console.WriteLine(c);

От типа данных зависит, какая информация будет хранится в переменной. Также от типа переменной зависит поведение операций над ними. Например, для строк знак плюс **(+)** означает склеивание между собой, а для целых и вещественных чисел, плюс складывает их и возвращает результат:

int a = 5;  
int b = 10;  
Console.WriteLine(a + b); // 15  
  
string s1 = "5";  
string s2 = "10";  
Console.WriteLine(s1 + s2); // 510

﻿Еще пример:

string name = "Олег";  
int age = 25;  
double weight = 80.5;  
string info = name + ", возраст " + age + ", вес " + weight;  
Console.WriteLine(info);