

Основы .NET разработки

Тема 1. Классификация языков программирования. Структура программы на C#. Первая программа

C#



Классификация языков программирования

- Язык программирования — формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — ЭВМ) под её управлением

Машинный код

Высокоуровнев

Низкоуровнев

ые

Сверх

Классификация языков программирования

- Уровень языка зависит не от его «крутости», а от того на сколько близко его команды похожи на команды процессора.
- Самые процессорные языки это машинный код, затем низкий уровень, высокий уровень, сверхвысокий уровень и обычный человеческий язык

Классификация языков программирования

машинный код

- Машинный код — это команды понятные компьютеру, состоящие из нулей и единиц

```
1010100101010100101010101010101010101010101
01010101010101001010101010100101011
01001010100010010100100101010010100
00011010000011101010000010010000000
11100110110101010011100110100111111
```



Классификация языков программирования

Низкоуровневые

- Низкоуровневые языки программирования, близкие к программированию непосредственно в машинных кодах используемого реального или виртуального процессора. Для обозначения машинных команд обычно применяется мнемоническое обозначение. Это позволяет запоминать команды не в виде последовательности двоичных нулей и единиц, а в виде осмысленных сокращений слов человеческого языка. Примеры языков: **Ассемблер, Forth,**

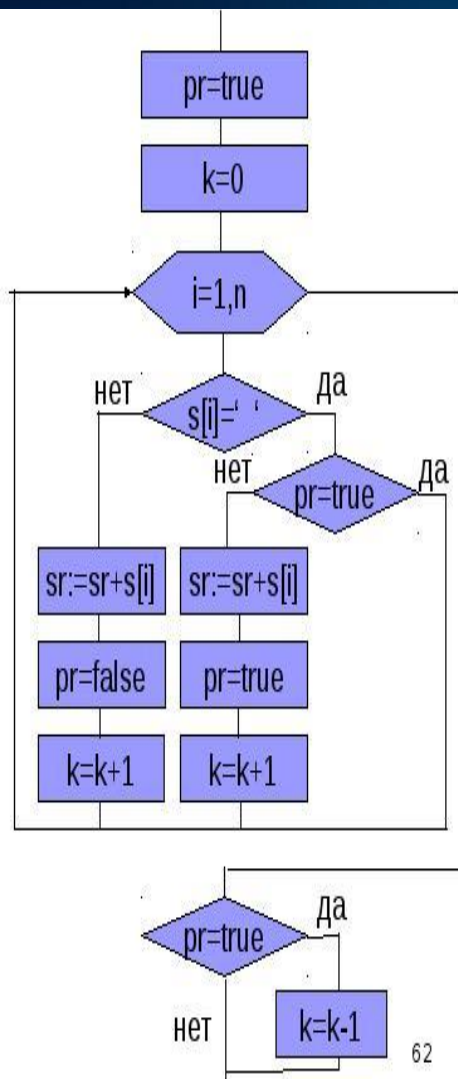
C

Классификация языков программирования

Низкоуровневые

```

mov     DL, 0
jcxz    prod3
mov     BX, 1
cld
cyc1:   lodsb
        cmp     AL, ' '
        je      prod1
        mov     BX, 0
        inc     DL
        stosb
        jmp     prod2
prod1:   cmp     BX, 1
        je      prod2
        mov     BX, 1
        inc     DL
        stosb
prod2:   loop    cyc1
    
```



Синтаксис языка Ассемблера

Пример программы в машинном коде



Assembler - EXAMPLE3.ASM

| File | Assembler | Instruction Set |
|---------------------|-----------|-------------------|
| .begin | | |
| in counter | | 11010000000001000 |
| while: load counter | | 00000000000001000 |
| compare zero | | 01110000000001001 |
| jumpeq done | | 1010000000000111 |
| out counter | | 11100000000001000 |
| decrement counter | | 01100000000001000 |
| jump while | | 10000000000000001 |
| done: halt | | 11110000000000000 |
| counter: .data 0 | | 00000000000000000 |
| zero: .data 0 | | 00000000000000000 |
| .end | | |

Классификация языков программирования

Высокоуровневые

- Высокоуровневый — язык программирования, разработанный для быстроты и удобства использования программистом. Основная черта высокоуровневых языков — это абстракция, то есть введение смысловых конструкций, кратко описывающих такие структуры данных и операции над ними, описания которых на машинном коде (или другом низкоуровневом языке программирования) очень длинные и сложны для понимания.

Классификация языков программирования

Высокоуровневые

```
.model small
.stack 256
.data
filda dw 1
fildb dd 1.5
.code
_main proc
PUBLIC main
EXTRN @cproc$CIF:proc
;EXTRN cproc:proc
mov ax,@data
mov ds,ax
push fildb
push filda
call @cproc$CIF
;call cproc
add sp,6
.exit 0
_main endp
end _main
```

```
#include <iostream.h>

void cproc(int x,float y)
{
cout << "Rez " << x << " ";
cout << y << "\n";
}
```



Классификация языков программирования

Сверх высокоуровневый

- Сверх высокоуровневый язык программирования (англ. very high-level programming language, VHLL) — язык программирования с очень высоким уровнем абстракции. В отличие от языков программирования высокого уровня, где описывается принцип «как нужно сделать», в сверх высокоуровневых языках программирования описывается лишь принцип «что нужно сделать»..

Python, Ruby, Haskell, Perl, мини язык AWK



Классификация языков программирования

Высокого уровня

Языки программирования



Компилятор Интерпретируем

Компилятор — программа, выполняющая преобразование файла с исходным кодом программы в исполняемый файл

Компиляция — преобразование программы, составленной на исходном языке высокого уровня в эквивалентную программу на низкоуровневом языке (машинном коде).

Интерпретатор — это программа, которая получает исходную программу и по мере распознавания конструкций входного языка реализует действия, описываемые этим и

Парадигмы программирования

Парадигма программирования — это совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию)

Парадигмы программирования

Императивн
ое



Структурн
ое



ОО
П



АО
П



Функциональн



Айтиландия

страна талантов

Императивное программирование процедурное

Программирование, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка. Выполнение программы сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти, то есть значений исходных данных, в заключительное, то есть в результаты.

Структурное программирование

Парадигма программирования, в основе которой лежит представление программы в виде иерархической структуры блоков.

В соответствии с парадигмой, любая программа строится без использования оператора **goto** из трёх базовых управляющих структур: **последовательность, ветвление, цикл**

Принципы структурного программирования

- Следует отказаться от использования `goto`
- Три базовые управляющие инструкции: последовательность, ветвление, цикл
- Вложенность управляемых конструкций
- Повторы оформлять в процедуры и функции
- Каждую логически законченную группу инструкций следует оформить как блок.
- Конструкции должны иметь один вход и один выход
- Подход «сверху вниз» «снизу вверх»



ООП программирование

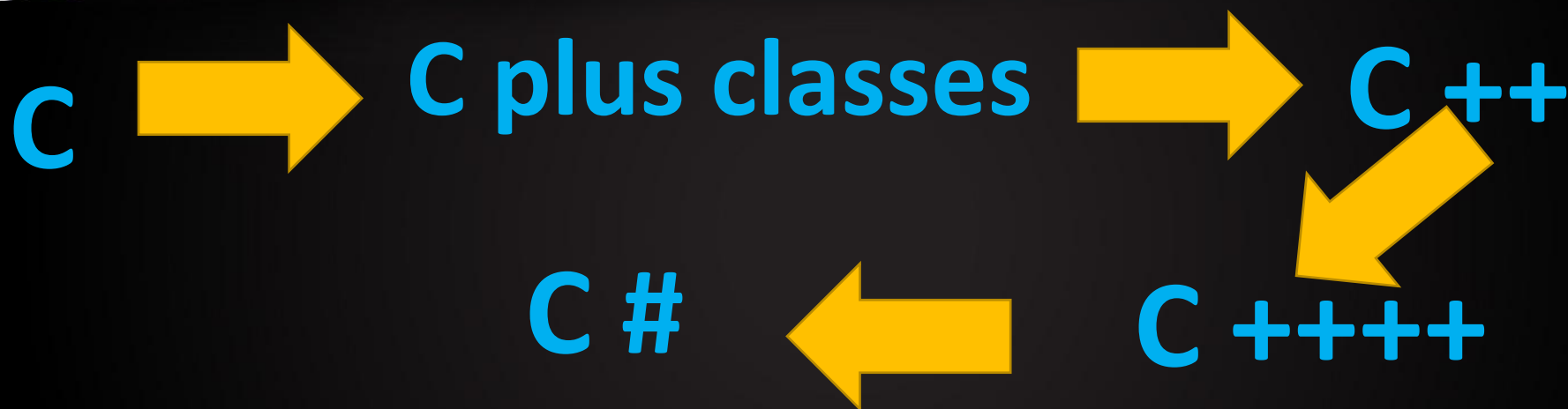
Объектно-ориентированное программирование (ООП) — парадигма (методология) программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Базовые принципы ООП

- Абстрагирование
- Инкапсуляция
- Полиморфизм
- Наследование
- Объект
- Класс



C#



Перед тем как изучать любой язык будь то русский, белорусский, английский и т.п. нужно изучить его алфавит. В языке C# также есть свой алфавит.

Алфавит с#

• A..Z a..z (_) как буква РУБАЕТСЯ

• Все цифры (0..9)

• ! " @ # % ^ & * () + - / \ | { } ~ [] : ; < > = ,

• //однострочные /*многострочные*/

Лексемы C#

- Идентификаторы
- Ключевые слова
- Знаки и символы операций
- Литералы
- Разделители



Идентификаторы

- Идентификаторы — имя чего-либо, состоящее из последовательности символов.
- В языке C# идентификаторами являются:
 - типы данных, имена переменных, функций, классов, интерфейсов, делегатов;
 - Могут состоять из букв (A..Z a..z) цифры (0..9) и _
- **Нельзя начинать с цифры, прописные и строчные буквы это разные символы не должны совпадать с ключевыми (зарезервированными) словами.**

Ключевые слова

| | | | | | |
|----------|-----------|-----------|------------|-----------|----------|
| abstract | as | base | null | operator | break |
| byte | case | catch | override | private | char |
| checked | class | const | public | ref | continue |
| decimal | default | delegate | sbyte | short | do |
| double | else | enum | stackalloc | string | event |
| explicit | extern | false | switch | throw | finally |
| fixed | float | for | try | uint | foreach |
| goto | if | implicit | unchecked | ushort | in |
| int | interface | internal | void | while | is |
| lock | long | namespace | object | new | return |
| params | sealed | this | unsafe | out | sizeof |
| readonly | static | typeof | volatile | protected | struct |
| true | ulong | using | bool | virtual | |

Знаки и символы операций

Литералы

+ - ++ -- * / = == != < <= > >= += -= *= /= % ||

- Литералами называют представление значения некоторого типа данных;
- Разделители (пробелы, табуляторы, переход на новую строку).

Структура программы на C#

- Подключение/объявление пространства имен (своего рода контейнера);
 - объявление класса
 - методы класса (подпрограммы), как минимум метод Main;
 - операторы и выражения;
- В каждом из этих блоков могут присутствовать комментарии — участок кода, невосприимчивый компилятором.

Структура программы на С#

```
1  //подключение пространства имен
2  using System;
3      using System.Collections.Generic;
4      using System.Linq;
5      using System.Text;
6  //объявление пространства имен
7  namespace ConsoleApp1
8  {
9      //объявление класса
10     class Program
11     {
12         //методы класса
13         static void Main(string[] args)
14         {
15             //тело метода операторы, выражения
16         }
17     }
18 }
```



Практическая часть

| № | Задание |
|---|--|
| 1 | Задание написать программу вывода строки “Hello world” на экран. сделать это при помощи обычного текстового редактора и компилятора C# |
| 2 | Задание написать программу вывода строки “Hello world” на экран.сделать это в любой из IDE (среды разработки), например VisualStudio |



практическая часть решение1

Набираем текст в любом текстовом редакторе и сохраняем его с расширением .cs под любым именем (program.cs) на любой из дисков в любую папку например F:\Айтиланд

```
1  using System;
2
3  namespace HelloWorld
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("Hello World!!!");
10             Console.ReadLine();
11         }
12     }
13 }
```



практическая часть решение1

Далее заходим в командную строку (win+R→cmd→enter) и переходим в папку где находится наш документ

```
C:\WINDOWS\system32\cmd.exe
```

```
Microsoft Windows [Version 10.0.17134.523]
```

```
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.
```

```
C:\Users\Сергей Николаевич>f:
```

```
F:\>cd F:\Айтиландия\C#\console
```

```
F:\Айтиландия\C#\console>
```

практическая часть решение1

Далее прописываем путь где может находится компилятор для этого в поиске на диске C напишите csc.exe это и есть компилятор и компьютер вам выдаст путь к нему обычно это тут (C:\Windows\Microsoft.NET\Framework\v3.5) впишите путь к компилятору в кавычки и через пробел пропишите имя вашего файла который нужно скомпилировать.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.523]
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

C:\Users\Сергей Николаевич>f:

F:\>cd F:\Айтиландия\C#\console

F:\Айтиландия\C#\console>"C:\Windows\Microsoft.NET\Framework\v3.5\csc.exe" programm.cs
Компилятор Microsoft (R) Visual C# 2008 версии 3.5.30729.8931
для Microsoft (R) .NET Framework версии 3.5
(c) Корпорация Майкрософт (Microsoft Corp.). Все права защищены.
```



практическая часть решение1

После компиляции у вас появится второй файл с таким же названием но с расширением .exe

```
Содержимое папки F:\Айтиландия\C#\console
21.01.2019  14:40    <DIR>          .
21.01.2019  14:40    <DIR>          ..
20.01.2019  13:23                293 programm.cs
21.01.2019  14:40                3 584 programm.exe
                2 файлов                3 877 байт
                2 папок  76 343 365 632 байт свободно
```

И теперь можно запустить эту программ, прописав название документа .exe

```
F:\Айтиландия\C#\console>programm.exe
Hello World!!!
exit
```



практическая часть решение2

Загрузить любую IDE (например VisualStudio) создать проект

и написать данный код

```
7      static void Main(string[] args)
8      {
9          Console.WriteLine("Hello World!!!");
10         Console.ReadLine();
11     }
```

Нажать запуск или F5 и посмотреть что получилось



Домашнее задание

| № | Задание |
|---|--|
| 1 | Установить любую IDE для работы C# предпочтительней VisualStudio |
| 2 | Написать программу вывода строки на экран (меня зовут ФИО. Я начинаю изучать C#) |

Спасибо за
внимание