.Константи и променливи

В C# константите могат да бъдат от произволен тип, който е определен от тяхното представяне. За да няма трудността при определяне на типа на една константа, в C# съществуват специални правила. Типът на целите числа е най-малкият цял тип, започвайки от int, който може да съхрани нейната стойност. Следователно в зависимост от стойността, типът на цялата константа може да бъде int, uint, long, ulong. Константите с плаваща точка са от тип double. Типът на константите по подразбиране може да бъде променен явно чрез суфикс. За да се зададе тип long на константа се добавя към нея “l”/”L”. Например 20L или 20l. За да се зададе тип uint – “u”/”U” – 100U (100u), за да се зададе ulong – “ul”/”UL”- 9 415UL. За float се добавя “f”/”F” – 10.15F. За decimal – “m”/”M” – 9.95M.

Целите константи с тип по подразбиране могат да бъдат присвоени на променливи от тип byte/sbyte/short/ushort, ако тяхната стойност е в допустимите стойности на съответния тип. Цяла константа int, uint винаги може да бъде присвоена на long. Hexadecimal започват с 0x (oxFF = 255). В C# няма осмични константи (в C++ има), тъй като те се използват рядко. Символната константа е единичен символ заграден в апострофи. Може да е и буква от Кирилицата. За някой често се използват неграфични символи, а за служебните символи се ползват следните еднобуквени означения:

a – system beep

b – символа “backspace”

f – за минаване на нова страница

n – нов ред

r – минаване в началото на същия ред

t – хоризонтална табулация

v – вертикална табулация

 – нулев байт

’ – символа „’”

” – символа „””

{ и } – съответно „”{ и „}”

\ - символа „\”

Константа от тип низ от символи е последователност от символи заградена в кавички. Такива константи се използват често в метода WriteLine, т.е. могат да съдържат управляващи символи, които при извеждане се изпълняват. Допълнително в C# може да се започва с „@”, след което следва низ в кавички. Съдържимото на низовата константа се използва без модификация и може да включва два или повече реда, следователно в този случай могат да се използва управляващи символи за нов ред, табулация и т.н. без да се използват еднобуквените escape означения, но ако трябва да се изведат кавички се задават 2 кавички една до друга.

Console.WriteLine(@“Това е копираща низова константа

която включва

няколко реда”);

Ще се изведе на нов ред. Копиращите низови константи се извеждат в програмата така както са били въведени.

тип списък_от_имена_на_променливи;

„Тип” е произволен тип, включително тези типове, разгледани до сега. Типът определя колко бита се разпределят за стойността на променливата и как се представя тази стойност в C#. Всяка променлива трябва да се дефинира преди да се използва. Това е необходимо тъй като компилатора трябва да знае типа на променливата при компилация на израз, който я използва. Преди да се използва променлива на нея трябва да и се присвои стойност. Това става по два начина:

- с отделна операция след дефинирането и;

- при дефиницията и.

Тогава дефиницията има следния общ вид:

тип променлива = израз;

Стойността на израза която се присвоява на променливата трябва да е съвместима с типа и.

Пример:

Пресмятане на обема на цилиндър при зададени радиус и височина.

double radius=4, height=5;

double volume = 3.1416*radius*radius*height;

В израза за инициализация т.е. за присвояване на начална стойност, може да се използва всеки елемент, който преди това вече е бил обявен. Такъв елемент може да е метод, друга променлива или константа.

Както казахме, блокът е последователност от дефиниции и оператори затворени в {}. Той задава областта на видимост като определя възможността за достъп до променлива и нейния живот. Областта на видимост се задава с блока. Променливите могат да се дефинират на произволно място. В блока променливата се създава при дефинирането й и се унищожава в края на блока. Областта на видимост може да се влага – при всяко създаване на блок се създава вложена област като външната област е видима за вътрешната, но обекти от вътрешната не са видими за външната.

Най-важни области на видимост са определени от тези на класа и методите. Тялото на метода е блок. Параметрите на блока имат област на видимост блока – тя е защитена от достъп от извън блока – това е едно от правилата, осигуряващи инкапсулация.

Пример:

using System;

```
class ScopeDemo

public static void Main()

int x; //известна в метода Main()

x = 10;

if (x == 10)

{ //видима е в метода Main()

int y = 20;

x = y * 2; //y е видима само вътре

y = 100; //грешка

Console.WriteLine("x=" + x);
```

Ако променливата се инициализира при нейното дефиниране то тя ще се инициализира всеки път при влизане в блока в който е дефинирана

Пример:

```
using System;

class VarIniDemo

public static void Main()

int x;

for (x = 0; x

int y = -1; //y се инициализира всеки път при влизане в блока

Console.WriteLine("y=" + y);

y = 100;

Console.WriteLine("y=" + y);

//y=100 се губи при всеки край на интерпретацията в цикъла
```

В C# името на променлива, дефинирана във вътрешна област на видимост, не може да съвпада с името на променлива, дефинирана във външната област на видимост. В C/ C++ няма такова ограничение.