Скаларни типове данни

1. Обща класификация

Езикът C++ има вградени типове данни, които се поддържат в ядрото на езика. Те са два вида: скаларни и съставни. На програмиста се дава възможност да дефинира и собствени типове при възникване на необходимост. Вградените скаларни типове данни са: целочислен, реален, символен и булев тип.

Припомни си!!!

Тип на данните се нарича категория данни, която има следните характеристики:

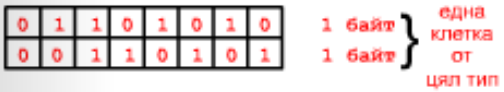
- *множество от допустими стойности;*
- *множество от допустими операции;*
- *множество от отношения (релации) между елементите на типа;*
- *множество от стандартни функции за типа (ако има такива).*

2. Целочислен тип

Целочисленият тип данни е предназначен за обработка на цели числа. Запазената дума за деклариране на променливи от този тип в *C++* е `int`.

Множеството от допустими стойности зависи от компилатора. В средата за програмиране *Borland C* диапазонът е от *-32768* до *32767*. А в *Microsoft Visual C++* диапазонът е от *-2147483648* до *2147483647*. За по-голяма яснота в курса ще разглеждаме диапазона, използван в *Borland C*.

Количеството памет за една променлива или константа от тип `int` е 2B.



Стойности: от -32768 до 32767

В таблицата са дадени различните модификатори, които може да се използват, и как те променят типа в *Borland C*.

Име	Диапазон	Памет
char, signed char	-128 до 127	1B
unsigned char	0 до 255	1B
short, sighed short	-32768 до 32767	2B
unsigned short	0 до 65535	2B
int, signed int	-32768 до 32767	2B
unsigned, unsigned int	0 до 65535	2B
long, signed long	-2147483648 до 2147483648	4B
unsigned long	0 до 4294967295	4B

2.1. Аритметични операции

Операциите над даден тип се извършват посредством оператори, които се изпълняват над даден операнд или операнди.

Унарни операции – извършват се над един операнд с операторите + и –, записани преди операнда. Те имат същото действие, както и в математиката – да потвърдят или да променят знака на дадено число.

Бинарни операции - извършват се точно над два операнда с операторите +, –, *, / и %, записани между операндите.

Два последователни аритметични оператора може да се допуснат, само ако единият от тях е унарен.

Оператори за:	Означение:	Примери:	
събиране	+	24+7→31	-32+86→54
изваждане	-	8-4→4	322-421→-99
умножение	*	5*4	
целочислено деление	/		
остатък при целочислено деление	%		

5*4=20

```
int a=2,b=3,c=4;
a=+b;a=-b;//унарни операции
//бинарни операции:
c=25+4;
c=25-4;
```

a	b	c
-3	3	29

2.2. Изрази с аритметични оператори

В математическите изрази операциите имат приоритет и така се знае кое действие се извършва първо.

В езиците за програмиране проблемът е решен по същия начин – всеки оператор си има приоритет и така се задава кое действие се извършва най-напред.

Ако разгледаме израза 5+6*3 в математиката, резултатът от изчислението ще бъде 23. Това е така, защото операцията умножение има по-висок приоритет от събирането, поради което първо се извършва умножението, а след това събирането.

Приоритет на операторите в низходящ ред:

най-висок `()`

`+, -` (унарни)

`*, /, %`

най-нисък `+, -` (бинарни)

За смяна на приоритета се използва операторът скоби `()`. Първо се изчислява изразът, записан в скобите. Ако има няколко израза в скоби, те се изчисляват в реда на задаването им отляво надясно.

*Пример:

Изразът (5+6)*3=33 е различен от 5+6*3=23

(3+6)/(8-5)=3 е различен от 3+6/8-5=3+0-5=-2

2.3. Оператори за сравнение

Операторите за сравнение, които се използват в езика C++, са дадени в следната таблица:

Оператор	Описание
<	по-малко
<=	по-малко или равно
>	по-голямо
>=	по-голямо или равно
==	равно
!=	различно

Като резултат от сравнението се връща логическа стойност *1* или *0* (*true или false*).

```
cout<<(5>4);          5>4 - истина
cout<<(-2<-20);    -2<-20 - лъжа
cout<<(6==6);         6==6 - истина
cout<<(6>=6);         6>=6 - истина
cout<<(5>=6);         5>=6 - лъжа
```

2.4. Вградени функции

За целочислен тип данни е вградена функцията `abs(x)`, която връща абсолютната стойност на аргумента `x`. За да се използва тази функция, е необходимо в началото на програмата да се включи библиотеката `math.h`.

*Пример:

```
cout<<abs(-12); //извежда 12
int a=abs(12);
cout<<a;        //извежда 12
int b=-12;
b=abs(b);
cout<<b;        //извежда 12
```

***Задача*: Да се състави програма, която въвежда цяло трицифрено число и извежда на екрана сумата и произведението от цифрите му.**

Решение:

```
//program_6_1.cpp
#include<iostream.h>
int main()
{
 int x,a,b,c,p,s;
 cout<<"x=";
 cin>>x;
 a=x/100;
 b=x/10%10;
 c=x%10;
 s=a+b+c;
 p=a*b*c;
 cout<<"sum="<<s<<endl;
 cout<<"pr="<<p<<endl;
 return 0;
}
```