

Упражнение 3

Оператори за инкрементиране и декрементиране.

- **Префиксни** оператори **++** и **--**

- Към променливата се добавя/изважда единица, след това се изпълнява командата / изчислява израз в която/който участват.

- *Пример:*

```
int num = 5;           // Увеличава променливата num с 1
cout << ++num;         // след това я извежда в конзолата.
                        // (Извежда 6 в конзолата)
```

- **Постфиксни** оператори **++** и **--**

- Изпълнява се командата в която участват, след това към променливата се добавя/изважда единица

- *Пример:*

```
int num = 5;           // Извежда на конзолата променливата
cout << num++;          // num след това я увеличава с 1
                        // (Извежда 5 в конзолата, но след това
                        // num ще бъде 6)
```

// Забележка: Същото се отнася и за оператора **--**. Разликата е, че
// **декрементира** (намалява) променливата с 1.

Съкратени записи на оператори с присвояване.

(+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)

a (знак)= b ⇔ a = a (знак) b

Пример: a += b; ⇔ a = a + b;

Важна разлика между операторите.

int num = 5;		int num = 5;
cout << num + 2; // 7		cout << num += 2; // 7
cout << num; // 5		cout << num; // 7

Сравняване на числа с плаваща запетая.

Сравнението на числа с плаваща запетая чрез оператора **==** най-вероятно **няма** да доведе до желания резултат!

Сравнението на такива числа става с **точност** до определена цифра след десетичната запетая.

Например: `const double EPS = 0.00001;`

Като за да сравним двете числа използваме изрази:

```
abs(num1 - num2) < EPS // Сравняваме дали числата num1 и
                        // num2 са равни (с точност EPS).
                        // abs(x) е функция от библиотеката
                        // math.h връщаща модула
                        // (абсолютната стойност) на
                        // подаденото число. abs(-5) == 5
```

Тернарен оператор.

<условие> ? <стойност, ако е **истина**> : <стойност, ако е **лъжа**>

Примери:

- `int max = a < b ? b : a;` // Присвояване на по-голямото от две
// числа.
- `int a = -5;` // Реализация на модул (абсолютна
`int mod = a < 0 ? -a : a;` // стойност) чрез оператора ?:

Условен оператор if-else.

Синтаксис:

```
if (<условие>)
{
    // <тяло> код ако условието е изпълнено.
}
else // не е задължителен компонент.
{
    // <тяло> код ако условието не е изпълнено.
}
```

Примери:

```
int num;
cin >> num;
if (num < 0)
    cout << "Negative" << endl;
else
    cout << "Non-negative" << endl;
// Ако въведеното число num е отрицателно изпълнението на
// програмата ще продължи в тялото (блока) на if-а и ще прескочи
// изпълнението на тялото на else-а.
// Ако е неотрицателно - ще прескочи тялото на if-а и ще продължи
// изпълнението си в тялото на else-а.
// Забележка: Фигурните скоби не са задължителни когато
// изпълняваме само една "команда" в if-а и else-а.

unsigned absValue;
if (num < 0) {                // Фиг. скоби отново не са задължителни,
    absValue = -num;         // но се слагат за да е по-лесно четимо.
}
cout << "The absolute value of num is " << num;
```

Оператор **switch-case**.

Синтаксис:

```
switch (<селектор>) {        // селекторите най-често са променливи
    case <константна стойност / литерал>:
        // Код изпълнен ако стойността на
        // селектора отговаря на този случай.
        break;              // спира изпълнението на последващите случаи.
    case <константна стойност / литерал>:
        // Код...
        break;
    // ... Още случаи ...
    default:
        // Ако стойността на селектора не отговаря на нито
        // един от случаите се изпълнява default случая.
        break;
}
```

Магически числа и защо не са ОК.

Числа в кода без никакво обяснение защо са там и какво означават. Объркват вас и четящия кода ви. За да се справите с тях ползвайте именувани константи.

Пример:

```
if (num1 + num2 < 7) { ... }
```

// 7 е магическо число без смисъл за четящия

// Правилният начин:

```
const int DAYS_IN_A_WEEK = 7;
```

```
if (num1 + num2 < DAYS_IN_A_WEEK) { ... }
```

Константни променливи.

Създаването на константни променливи става чрез ключовата дума `const`. Не можем да декларираме константна променлива без да ѝ зададем стойност! Константна променлива може единствено да има стойност, която се знае (може да се изчисли) по време на компилация. Имената на константи пишем само с ГЛАВНИ букви, като разделяме думите с долна черта, за да ги различаваме от неконстантните променливи.

Примери:

```
const int DAYS = 7; // Дефинираме константа от тип int със стойност 7
```

```
DAYS = 4; // Грешка: DAYS е константа (не можем да я променяме)
```

```
const int SOME_CONSTANT;
```

// Грешка: SOME_CONSTANT не е инициализирана със стойност

Изброен тип (Енумерация).

Синтаксис:

```
enum <идентификатор> {  
    CONSTANT_0 [= стойност],  
    CONSTANT_1 [= стойност],  
    ...  
}
```

```
<име на променлива>, <име на променлива 2>, ...;
```

Изброените типове се състоят от множество именувани константи. Всяка константа има стойност, като по подразбиране започват от 0 и всяка следваща има стойност с 1 по-голяма от предходната. След дефиницията на изброен тип стои **;**.

Пример:

```
enum Vegetables {           // Енумерация на име Vegetables
    KALE,                   // По-подразбиране има стойност 0
    CUCUMBER,               // Има стойност 1
    POTATO = 8,             // Зададена е стойност 8
    AUBERGINE               // Има стойност 9
} veg1 = Vegetables::CUCUMBER; // Дефинираме променлива от тип
                                // Vegetables на име veg1 със
                                // стойност Vegetables::CUCUMBER (1)
```

Често се ползва с оператора **switch** (07-Switch-Example.cpp). Изброеният тип може да се преобразува до целочислен тип и обратно.