Sam dla siebie C++
każdy programować może

**niedziela, dnia 07.09. 2025 r.**

# *Instrukcje w C++ nr 005/25*

## Instrukcje, sterujące, warunkowe, logiczne.

### Co to jest instrukcja i do czego służy?

Tworząc programy w języku C++, tworzę komendy którymi polecam dla komputera wykonanie operacji opisanej w tej komendzie. Program który tworzę, składa się z nie jednej komendy lecz z więcej niż jedna. Jedna taka komenda, jest to polecenie wykonania działania wyrażonego wyrażeniem zawartym w tej komendzie. W ten sposób wyrażonym wyrażeniem instruuję co komputer ma wykonać. Bowiem wyrażeniem przedstawiam sekwencję operatorów i operandów. Innymi słowy, powstała komenda instruująca wykonanie zadania dla komputera jest samodzielną, najmniejszą jednostka wykonywalną programu w języku C++ i określa się terminem **instrukcja**. Każdą instrukcję kończę postawionym na końcu komendy – średnikiem **;**

Operatory które używam do tworzenia programu wykazałem w temacie poprzednim. Natomiast operandy, są to elementy wyrażenia podające wielkość operacji reprezentowanej przez operatory.

Stąd też jak spostrzegłem, występują między innymi instrukcje warunkowe, logiczne, selekcji, wyboru, itp. itd.

Jak wiadomo, żeby było coś nazwane sterującym, to to coś musi właśnie sterować. Mianowicie musi pozwolić na kontrole przepływu wykonania programu, umożliwić na podjęcie decyzji a także umożliwić powtarzanie operacji. Czyli, podaje warunek który komputer musi spełnić. Tym sposobem powstają typy instrukcji.

W związku z powyższym jak rozumiem są to **instrukcje warunkowe** do których należy:

### instrukcja if

zapis instrukcji if

```
if (warunek) {
        instrukcja; /* kod który zostanie wykonany jeśli warunek
                       jest prawdziwy */
}
```

Instrukcja if – jest instrukcją spełniającą podany warunek. Jeśli ten podany warunek jest spełniony, następuje wykonanie czynności lub ciągu czynności , w przeciwnym wypadku pomija się daną czynność lub ciąg czynności. Warunek ujmuje się w nawiasy okrągłe ( .... ) .

if – *oznacza jak słowo*: jeżeli.

Np.: Jeżeli wyciągniesz los w którym będzie podana wartość wygranej, dostaniesz wygraną.

Przykład prostej instrukcji if.

```
/* instrukcja if – program do bloga */

#include <iostream>
#include <string>
#include <conio.h>
#include <math.h>

    using namespace std;

   int main() {

   int a, b;

   if ( a > b )                              // podany warunek a > b

       cout << '' podaj liczbe a: '' << endl; // program prosi o liczbę a
       cin >> a;                              // podajesz z klawiatury liczbę a
       cin.ignore ();                         // zatrzymuje działanie

       cout << '' podaj liczbe b: '' << endl; // program prosi o liczbę b
       cin >> b;                              // podajesz a klawiatury liczbę b
       cin.ignore ();                         // zatrzymanie działania

              /* program wykonuje operacje, tj. bada czy podane liczby
                spełniają warunek  a > b i po zbadaniu warunku daje odpowiedź */

      cout << '' Czy podajac liczby wiedziales o jaki warunek chodzi? '' << endl;
      cout << '' Powinno być a > b.'' << endl;

   getch ():
   return 0:

}
```

Za liczbę a podam 7 a za liczbę b podam 3. Oto co się pojawi na monitorze.

```
podaj liczbe a
7
podaj liczbe b
3
Czy podajac liczby wiedziales o jaki warunek chodzi?
Powinno być a > b.
_
```

**instrukcja if - else**

zapis instrukcji if - else

```
if (warunek) {
        instrukcja1; /* kod który zostanie wykonany jeśli warunek
                        jest prawdziwy */
} else {
        Instrukcja2; /* kod który zostanie wykonany jeśli warunek
                        jest prawdziwy */
}
```

Instrukcja if – jest instrukcją pozwalającą spełnić podany warunek o ile podany warunek jest do spełnienia.. Jeśli ten podany warunek jest spełniony, następuje wykonanie czynności lub ciągu czynności w instrukcji1, w przeciwnym wypadku pomija się daną czynność lub ciąg czynności instrukcji1 i jest wykonywana instrukcja2. Warunek ujmuje się w nawiasy okrągłe ( .... ) .

if – *oznacza jak słowo*: jeżeli,

else – w przeciwnym wypadku

A to przykład instrukcji if – else

```
/* instrukcja if – else */

#include <iostream>
#include <math.h>
#include <conio.h>

   using namespace std;

  int main()  {

        int a, b;
        double c = a * b;

        cout << "Podaj liczbe a: " << endl;
        cin >> a;
        cin.ignore ();

        cout << "Podaj liczbe b: " << endl;
        cin >> b;
        cin.ignore ();

        cout << ''a * b = '' << a * b << endl;

        if ( a * b == c ) {
```

```
        cout << '' Wynik c należy do zbioru liczb równych 30.'' << endl
       } else {

      cout << '' Wynik c należy do zbioru liczb większychj niż 30.'' << endl

     }

getch ();
return 0;

}
```

Dla przykładu gdy z klawiatury za a ppdstawię liczbę 3 i za b podstawię liczbę 10 na monitorze pojawi się końcowy wynik programu wykonanego:

```
Podaj liczbe a
3
Podaj liczzbe b
10
a * b = 30
Wynik c = 30 należy do zbioru liczb równych 30.
```

Gdy dla przykładu z klawiatury za a podstawię 9 i za b podstawię 8 , wtedy na monitorze pojawi się końcowy wynik programu wykonanego:

```
Podaj liczbe a
9
Podaj liczbe b
8
a * b = 72
Wynik c należy do zbioru liczb większych niż 30.
```

Więc jak widać w czasie wykonywania programu, zostały wykonane wszystkie instrukcje spełniające podany warunek .

**instrukcja if – else if**

zapis instrukcji if – else if

```
if (warunek1) {
        instrukcja1; /* kod który zostanie wykonany jeśli warunek
                        jest prawdziwy */
} else if (warunek2{
        instrukcja2; /* kod który zostanie wykonany jeśli warunek
                        jest prawdziwy */
} else {
    instrukcja1; /* kod który zostanie wykonany jeśli żaden z warunków
                    nie jest prawdziwy */
```

Przykład programu if – else if.

```
/* instrukcja if – else if , wiek – co mozna zrobic, */

#include <iostream>

#include <conio.h>

    using namespace std;

    int main( ) {

   int wiek;

   cout << '' podaj wiek: '' << endl;
   cin >> wiek;
   cin.ignore ( );

  if ( wiek >= 65 ) {

  cout << " Ta osoba jest już emerytem." << endl;

  } else if ( wiek >= 35 ) {

  cout << " Ta osoba jest dorosła i może być admirałem." << endl;

  } else if ( wiek >= 20 ) {

  cout << " Ta osoba jest dorosła." << endl;

  } else if ( wiek >= 20 ) {

  cout << " Ta osoba jest dorosła." << endl;
```

```
   } else {

    cout << "Ta osoba jes nastolatkiem" << endl;

   }

getch ( );
return 0;
}
```

Powyższy program jest programem przykładowym którym komputer może dać odpowiedź gdy się osoba zapyta co może w danym wieku robić. Jest to program wielowariantowy, jest wiele postawionych warunków i z tak wielu warunków dostaję odpowiedź tylko na ten warunek, który mnie interesuje.

Oto wyniki działania powyższego programu.

Otwieram powyższy program i widzę:

```
Podaj wiek
-
```

Następnie podaję z klawiatury wiek, np.: 17 i klikam w enter. Otrzymuję to co widzę na monitorze:

```
Podaj wiek:
17
Ta osoba jest nastolatkiem.
-
```

Gdy powtórzę czynność i z klawiatury podam inny wiek, np.: 45, otrzymam następujący efekt:

```
Podaj wiek:
45
Ta osoba jest dorosła i może być admirałem.
-
```

A co się stanie gdy po otworzeniu w/w programu podam z klawiatury wiek określony liczbą 30?

Oto efekt:

```
Podaj wiek.
30
Ta osoba jest już dorosła.
-
```

Pięknie prawda?

No właśnie, a jeszcze spróbuję wiek określić liczbą 75. Zobaczę jaki efekt osiągnę;

```
Podaj wiek.
75
Ta osoba jest już emerytem.
-
```

Nie mniej tworząc powyższy program , musiałem zachować logiczny porządek. Nie zachowując logicznego porządku w formacie powyższego programu, uruchomiony program, gdy uda się otworzyć, będzie programem wadliwym i nie będzie wykonywał prawidłowego wyboru – co mogę wykonać w danym wieku.

## instrukcja wyboru – switch

Instrukcja wyboru – switch, pozwala na wybór jednego z wieli bloków kodu w zależności od wartości wyrażenia.

Instrukcja wyboru-switch najpierw wykonuje czynność polegającą na obliczeniu wartości podanej w nawiasach okrągłych ( ) przy słowie switch, to jest zapisanego wyrażenia które musi być typu całkowitego, albo konwertowalnego do niego.

zapis instrukcji wyboru – switch:

```
switch ( wyrażenie ) {

        case wartość1:

        ..... // kod do  wykonania, jeśli wyrażenie == wartosc1

        break;


        case wartość1:

        ..... // kod do  wykonania, jeśli wyrażenie == wartosc1
```

```
break;

case wartość2:

. . . . . // kod do  wykonania, jeśli wyrażenie == wartosc2

break;

. . . . . .

default:

. . . . . // kod do  wykonania, jeśli żadna wartość nie pasuje

}
```

Następnie gdy jego wartość jest zgodna ze słów kluczowych case, to od tego miejsca wykonuje się instrukcja.

case wartość

Koniec wykonania instrukcji, następuje wówczas, gdy na drodze wykonywanych czynności, pojawi się instrukcja brek

break – jest używany do wyjścia z instrukcji switch po wykonaniu odpowiedniego bloku kodu.

Instrukcja break występuje po to , aby po podaniu jednej wartości case, nie następowało wykonanie wszystkich instrukcji na raz.

Instrukcja default jest instrukcją która w razie podania wartości nie pasującej do powyżej wskazanych etykiet, podaje informacje, że, na przykład: Podales blendna wartosc instrukcji. W tym programie brak takiej instrukcji.

default – jest opcjonalny i wykonuje się, gdy żadna z wartości nie pasuje do wyrażenia.

Oto przykładowy program zawierający instrukcje switch.

```
/* program wyboru – switch */

#include <iostream>

#include <conio.h>

    using namespace std;

    int main( ) {

    int ktory;

    cout << "Szefie ktory podzespol sprawdzic? " << endl;
    cin >> który;

    switch ( który ) {
```

```
case 10:
    cout << "Motor diesel S-148turbo. Poruwnaj z tabela PX-148DT" ;
    break;
case 12:
    cout << "Stery typ MORS-654m. Dane sa w tabeli MSM-654m." ;
    break;
case 34:
    cout << "Elektronike pokladowa. Informacja znajduje się w tabeli ELEC1924." ;
    break;
default:
    cout << "Podales blendny numer. Nie ma takiego w tym katalogu.";
    break;
```

Gdy otworzymy program, na monitorze zobaczymy:

```
Szefie ktory podzespol sprawdzić?
-
```

Po odczytaniu powyższego pytania podajemy o jaki numer podzespołu chodzi, a wiec podajemy numer podzespołu np.: 10. Co zobaczymy? Zobaczymy to co widać nic innego. Oto widok:

```
Szefie ktory podzespol sprawdzić?
10
Motor diesel S-148turbo. Poruwnaj z tabela PX-148DT
```

A gdy napisze numer 12? Będzie następujący efekt.

```
Szefie ktory podzespol sprawdzic?
12
Stery typ MORS-654m. Dane sa w tabeli MSM-654m.
```

Tak samo będzie gdy podam nr 34. Dla pewności sprawdzam. Oto co mamy…

```
Szefie ktory podzespol sprawdzic?
34
Elektronike pokladowa. Informacja znajduje się w tabeli ELEC1924.
```

A tak dla pewności podam błędny numer. To jest numer który nie znajduje się w programie, to jest numer 25. I jaki to będzie efekt? Proszę, oto efekt.

```
Szefie ktory podzespol sprawdzic?
25
Podales blendny numer. Nie ma takiego w tym katalogu.
```

I tym to sposobem opanowałem podstawowe instrukcje C++, które mi pozwoliły jako tako osiągnąć wiedzę o programowaniu w języku C++.

Jeśli chcesz sprawdzić moje wypociny - zapraszam. Może nawiążemy kontakt i poprowadzimy dialog o moich efektach nauki języka C++.

**Krótka uwaga.:**

Niniejsze opracowanie, jak i następne opracowania celem nauki własnej którą kontynuuję, opieram na lekturach takich jak:

1. Język C++ Szkoła programowania Stephen Prata
2. Programowanie w języku C++ Wiesław Porębski
3. Podstawy języka C++ Stanley B. Lippman
4. Symfonia C++ standard Jerzy Grębosz
5. PRZEWODNIK DLA POCZĄTKUJĄCYCH C++ ALEX ALLAIN
6. i internet