Podstawy programowania w C++

Instrukcje sterujące

Opracował: Andrzej Nowak

Bibliografia:

Nauka programowania dla początkujących; A. Struzińska-Walczak / K. Walczak

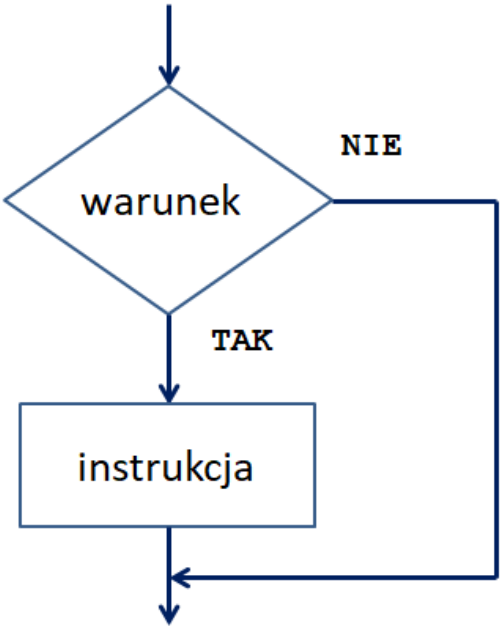
CPA: PROGRAMMING ESSENTIALS IN C++ https://www.netacad.com

Instrukcje sterujące służą do sterowania przebiegiem programu. Dzięki nim są podejmowane decyzje o wykonaniu tych czy innych instrukcji w programie. Decyzje te zależą od wyniku sprawdzenia określonych warunków, czyli od ustalenia, czy warunek jest prawdziwy czy nie.

Instrukcja warunkowa - if umożliwia kontrolę wykonywania poszczególnych instrukcji w obrębie programu w zależności od spełnienia warunku.

Instrukcja warunkowa może mieć dwie postaci:

Niepełna instrukcja warunkowa (prosta)



Schemat instrukcji warunkowej(niepełnej) – jeśli warunek jest spełniony instrukcja zostanie wykonana w przeciwnym wypadku wykonana zostanie następna instrukcja po instrukcji warunkowej

Zapis w C++

```
if (warunek)
{
    instrukcja;
}
```

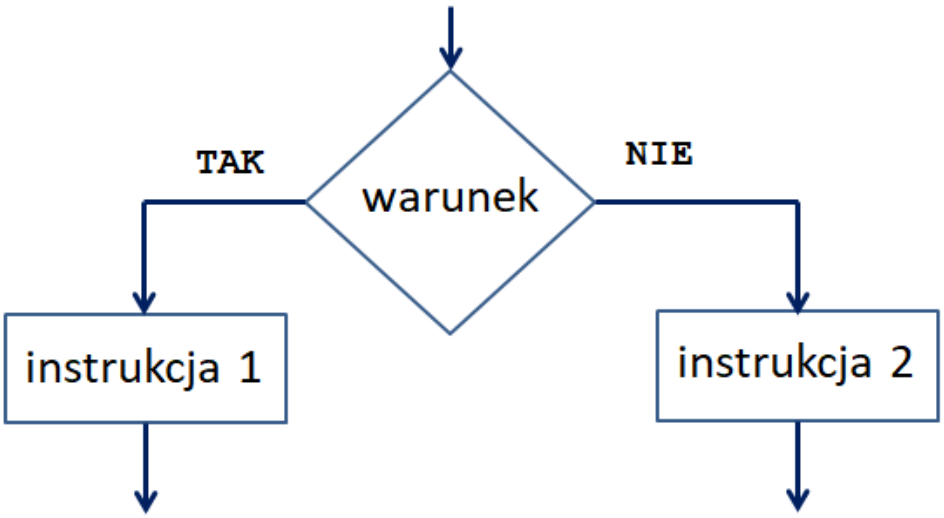
Program - przykład użycia

Napisz program obliczający pole kwadratu.

Pamiętaj, że długość boku każdej figury geometrycznej jest liczbą dodatnią. Program powinien sprawdzić czy podana przez użytkownika liczba ma wartość większą od zera. Jeśli tak, to zostanie obliczone pole kwadratu, w przeciwnym wypadku program się zakończy.

```
#include <iostream>

using namespace std;

int main()
{
      double a,p;
      cout << " podaj dlugosc boku kwadratu : ";
      cin >> a;

      if ( a > 0)
      {
            p = a * a;
            cout << "Pole kwadratu wynosi :  " << p;
      }
      return 0;
}
```

Pełna instrukcja warunkowa (instrukcja warunkowa z alternatywą)



Schemat instrukcji warunkowej(**pełnej**) – jeśli warunek jest spełniony zostanie wykonana instrukcja 1 w przeciwnym wypadku wykonana zostanie instrukcja 2

Zapis w C++

```
if (warunek)
{
    instrukcja 1;
}
else
{
    instrukcja 2;

}
```

Program - przykład użycia

Napisz program obliczający pole kwadratu.

Użycie pełnej instrukcji warunkowej w tym przykładzie poprawi działanie programu. Jeśli użytkownik poda ujemną wartość boku kwadratu , na ekranie zostanie wyświetlony komunikat o źle podanej wartości.

```
#include <iostream>

using namespace std;

int main()
{
    double a,p;
    cout << " podaj dlugosc boku kwadratu : ";
    cin >> a;

    if ( a > 0)
    {
        p = a * a;
        cout << "Pole kwadratu wynosi :  " << p;
    }
    else
    {
        cout <<"Dlugosc boku kwadratu musi być liczba dodatnia" ;
    }
    return 0;
}
```

Warunki proste i złożone

Program_1

Napisz program, który sprawdzi, czy dana liczba naturalna jest liczbą parzystą i niepodzielną przez 3. Zastosuj **warunki proste** w instrukcji **if**.

```
#include <iostream>

using namespace std;

int main()
{
	int a;
	cout << " podaj liczbe : ";
	cin >> a;

	if ( a %2 == 0)
	{
		if ( a %3 != 0)
		{
			cout << "Liczba  " << a << " jest parzysta niepodzielna przez
			3  " << endl;
		}
		else
		{
			cout << "Liczba  " << a << " nie jest jednocześnie liczba
			parzysta i niepodzielna przez 3  " << endl;
		}
	}
	else
	{
		cout << "Liczba  " << a << " nie jest jednocześnie liczba parzysta
		i niepodzielna przez 3  " << endl;
	}
	return 0;
}
```

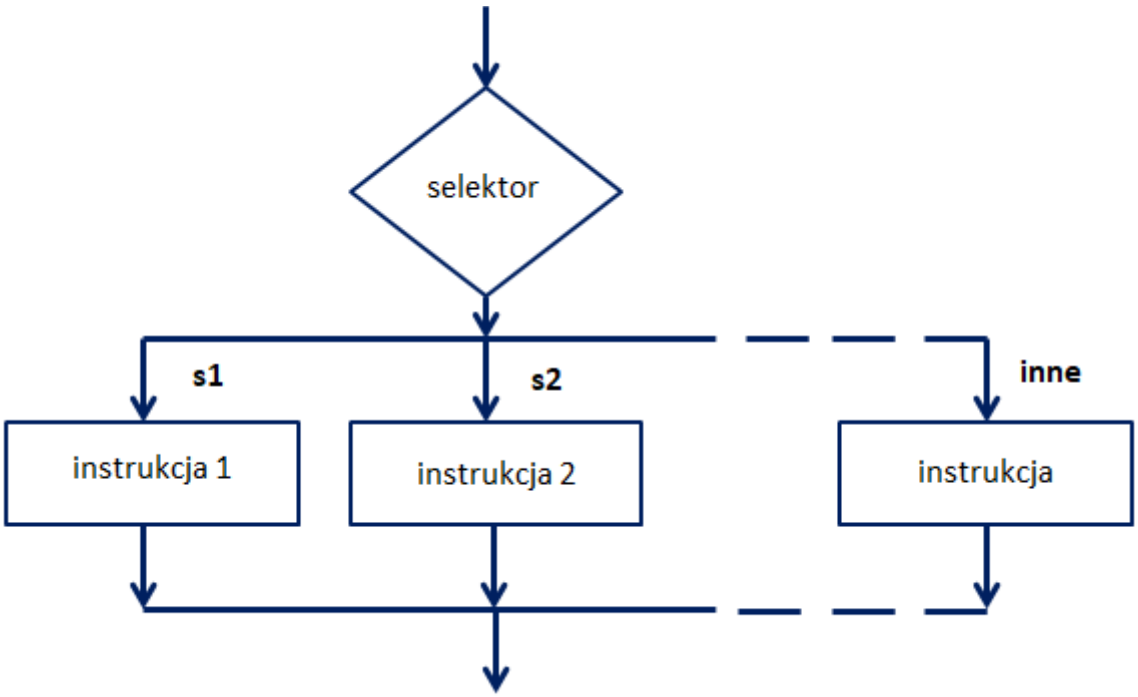
Program_2

Napisz program, który sprawdzi, czy dana liczba naturalna jest liczbą parzystą i niepodzielną przez 3. Zastosuj **warunki złożone** w instrukcji **if**.

```
#include <iostream>

using namespace std;

int main()
{
	int a;
	cout << " podaj liczbe : ";
	cin >> a;

	if ( a %2 == 0 && a %3 != 0)
	{
		cout << "Liczba  " << a << " jest parzysta niepodzielna przez 3  "
		<< endl;
	}
	else
	{
		cout << "Liczba  " << a << " nie jest jednocześnie liczba parzysta
		i niepodzielna przez 3  " << endl;
	}

	return 0;
}
```

Instrukcja switch

Instrukcja switch służy do podejmowania wielowariantowych decyzji



Schemat blokowy instrukcji switch

Zapis w C++

```
switch (selektor)
{
case s1:
        instrukcja 1;
        break;
case s2:
        instrukcja 2;
        break;
……..
default:
        instrukcja;
        break;
}
```

Selektor jest to zmienna, która może przyjmować różne wartości.

Opis działania instrukcji:

Jeśli wartość **selektora** odpowiada wartości podanej w jednej z etykiet **case**`(s1, s2,…)`, wtedy instrukcje są wykonywane począwszy od tej etykiety.

Wykonanie kończy się po napotkaniu instrukcji **break** – następuje wyjście z instrukcji **switch.**

Jeśli wartość **selektora** nie zgadza się z żadną z wartości podanych przy etykietach **case** to wykonają się instrukcje umieszczone po etykiecie **default** (może się znajdować w dowolnym miejscu instrukcji **switch**, ale zazwyczaj podaje się ją na końcu).

Instrukcja **break** powoduje natychmiastowe przerwanie instrukcji . Jeśli mamy do czynienia z kilkoma pętlami zagnieżdżonymi to instrukcja **break** powoduje przerwanie tylko tej pętli, w której bezpośrednio tkwi.

Program- przykład użycia

Napisz program, który:

- wypisze na ekranie komunikat ”Poniedziałek” , jeśli użytkownik wpisze liczbę 1;
- wypisze na ekranie komunikat ”Wtorek” , jeśli użytkownik wpisze liczbę 2;
- wypisze na ekranie komunikat ”Środa” , jeśli użytkownik wpisze liczbę 3;
- wypisze na ekranie komunikat ”Podałeś złą liczbę ” , jeśli użytkownik wpisze każdą inną liczbę.

```
#include <iostream>
using namespace std;

int main()
{
      int dzien;      //selektor
      cout << ” podaj cyfre od 1 do 3 : ”;
      cin >> dzien;

      switch ( dzien)
      {
      case 1:
            cout << ”Poniedzialek ”;
            break;
      case 2:
            cout << ”Wtorek ”;
            break;
      case 3:
            cout << ”Sroda ”;
            break;
```

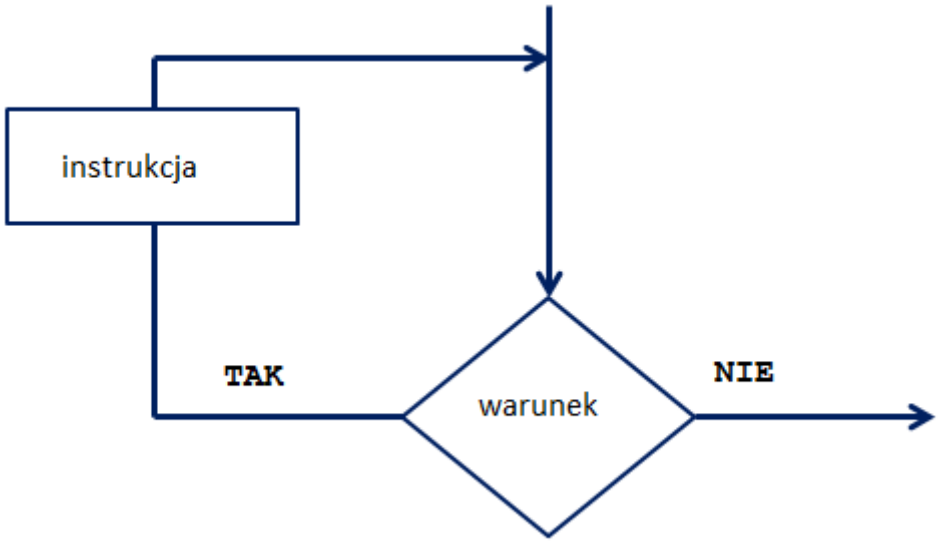
```
    default:
        cout << ”Podales zla liczbe ”;
        break;
    }
    return 0;
}
```

Pętla iteracyjna while

Pętla to konstrukcja programowania, która umożliwia cykliczne wykonywanie ciągu instrukcji określoną liczbę razy, do momentu zajścia pewnych warunków, dla każdego elementu kolekcji lub w nieskończoność.

W języku C++ do zbudowania pętli wykorzystuje się następujące instrukcje:

- **while** – wykorzystywana jeśli liczba powtórzeń ma zależeć od warunku pętli;
- **do …while** – wykorzystuje się jeśli dany blok instrukcji musi być wykonany przynajmniej raz;
- **for** – wykorzystuje się gdy konieczne jest określenie liczby powtórzeń danego bloku instrukcji



Schemat blokowy pętli while

Opis działania pętli:

Tak jak w przypadku instrukcji warunkowej , najpierw oblicza się wartość warunku.

Jeśli wynik jest zerowy, to następuje wyjście z pętli.

Jeśli wynik jest wartością niezerową , to jest wykonywana instrukcja (blok instrukcji), a następnie ponownie jest sprawdzana wartość warunku.

Pętla zakończy działanie jeśli warunek nie zostanie spełniony – i to jest jedyny powód przerwania jej działania.

Uwaga:

Pewnym niebezpieczeństwem jest warunek, który będzie zawsze prawdziwy – pętla może działać w nieskończoność.

Zapis w C++

```
while (warunek)
{
    instrukcja;
}
```

Program- przykład użycia

Napisz program wyświetlający na ekranie znaki podane z klawiatury do momentu , aż zostanie podany znak „k”

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main()
{
    char a;
    cout << ” program wyswietlajacy na ekranie znaki podane z klawiatury do
    momentu , az zostanie podany znak – k ”;

    cout << ”podaj znak”;
    cin >> a;

    while (a!= ‘k’)
    {
        cout << ” Podaj kolejny znak”;
        cin >> a;
    }
    cout << ” Podales ” << a << ”wiec koncze ”;

return 0;
}
```

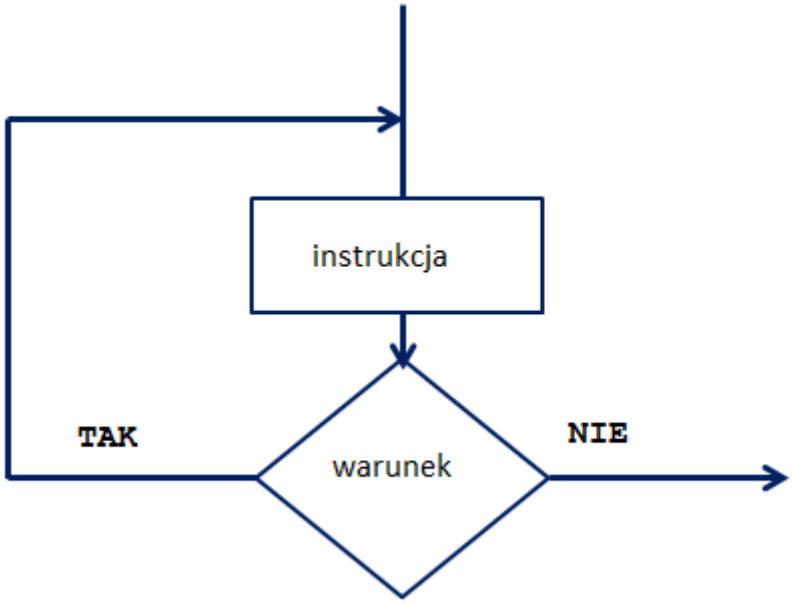
Program- przykład użycia

Napisz program wyznaczający największy wspólny dzielnik dwóch liczb podanych z klawiatury.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main()
{
    int a,b;
    cout << ” podaj liczbe a : ”;
    cin >> a;
```

```
    cout << ”podaj liczbe b:”;
    cin >> b;

    while (a!= b)
    {
        if (a < b)
        {
            b -= a;
        }
        else
        {
            a -= b;
        }
    }
    cout << ” Najwiekszy wspolny dzielnik wynosi : ” << a <<endl;
    return 0;
}
```

Pętla iteracyjna do …while



Schemat blokowy pętli do … while

Opis działania pętli:

Instrukcja, która znajduje się wewnątrz pętli zostanie wykonana przed sprawdzeniem warunku. Po wykonaniu pierwszy raz instrukcji zostanie sprawdzony warunek i jeśli będzie spełniony pętla będzie wykonywać kolejne powtórzenia instrukcji , ale jeśli nie będzie spełniony pętla się zakończy.

Zapis w C++

```
do
{
    instrukcja;
}
while (warunek;
```

Program- przykład użycia

Napisz program wyświetlający na ekranie zadaną liczbę gwiazdek

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main()
{
	int i = 0;
	cout << ” program wyswietlajacy na ekranie zadana liczbe gwiazdek”;
	cout << ”podaj liczbe liczbe gwiazdek do narysowania” << endl;
	cin >> i;

	do
	{
		cout << ”*”;
		i--;
	}
	while (i);

	cout<<endl;
	return 0;
}
```

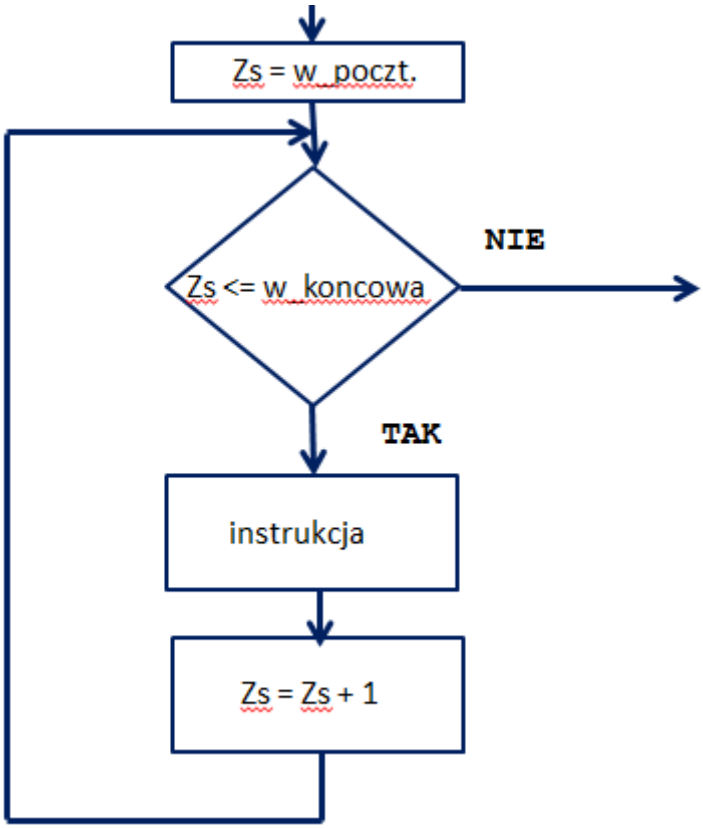
Program- przykład użycia

Napisz program, który wypisze ciąg malejący liczb całkowitych od 25 do 1 w postaci kolumnowej.

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main()
{
	int i = 25;

	do
	{
		cout << i<< endl;
		i--;
	}
	while (i >= 1);

	return 0;
}
```

Pętla iteracyjna for

Do wykonywania pętli o określonej liczbie powtórzeń stosuje się instrukcję for.

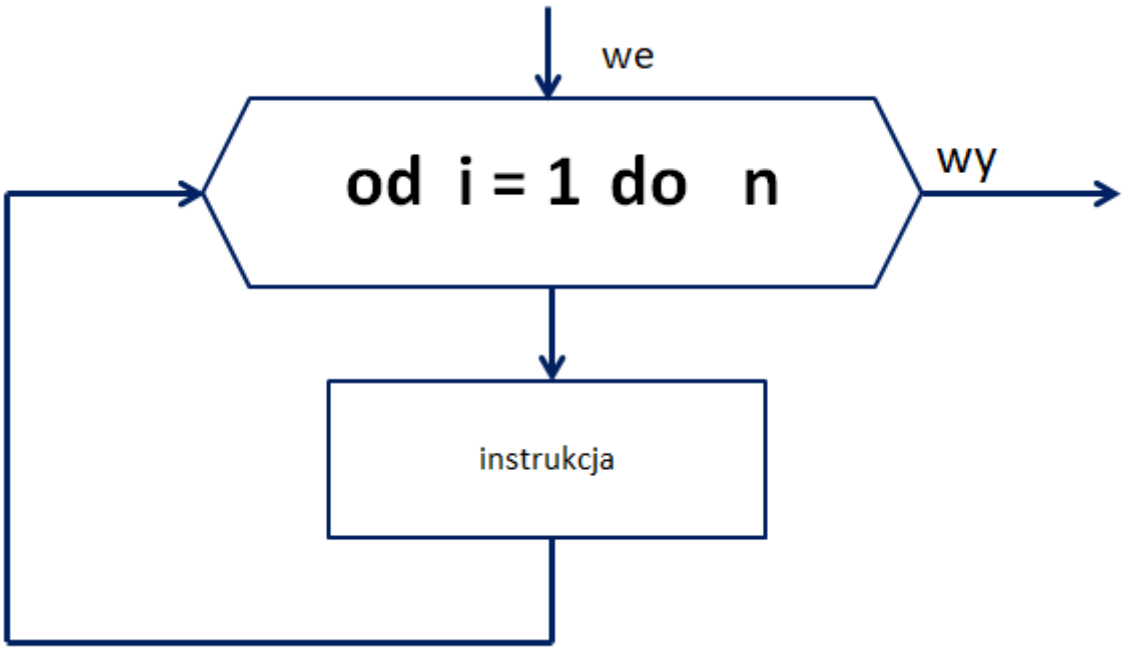


Schemat blokowy instrukcji for (wersja 1) – skomplikowane użycie w wypadku pętli zagnieżdżonych

Opis działania pętli:

Instrukcja ma trzy argumenty:

- Zs = w_pocz – określa wartość początkową zmiennej sterującej pętli - Zs
- Zs <= w_koncowa – wyznacza wartość końcową zmiennej sterującej
- Zs = Zs +1 – sposób zmiany wartości zmiennej sterującej (inkrementacja)



Schemat blokowy instrukcji for (wersja 2)

Opis działania pętli:

Wykonaj instrukcję dokładnie n – razy .

i= 1- wartość początkowa licznika wykonania pętli

n – wartość końcowa

Zapis w C++

```
for (Zs=w_pocz; Zs<=w_koncowa; Zs++)
{
    instrukcja;
}
for (i=1; i<=n; i++)
{
    instrukcja;
}
```

Program- przykład użycia

Napisz program tworzący tablicę jednowymiarową i wypisujący elementy tej tablicy.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    int n, tab[100];
```

```
    //interfejs
    cout << " program tworzacy tablice jednowymiarowa i wypisujacy elementy
    tej tablicy " << endl;
    cout<<
    "=============================================================================
    =====" << endl;
    cout << "podaj z ilu elementow ma skladac sie tablica " << endl;
    cout<<endl;
    cin >> n;
    cout<<endl;

    //tworzenie tablicy
    for (int i=0; i < n ; i++)
    {
         cout << "podaj wartosc elementu ";
         cin >> tab[i];
    }
    cout<<endl;

    //interfejs
    cout << "Elementy tablicy " << endl;
    cout<< "-------------------------------------" << endl;
    cout<<endl;

    //wypisywanie elementow tablicy

    for (int i=0; i < n ; i++)
    {
         cout << tab[i] <<" ";
    }
    cout<<endl;
    cout<<endl;

    return 0;
}
```