Pętla nieskończona

- Pętla nieskończona nie zawiera warunku kończącego lub nigdy nie jest on osiągnięty.

- Przykłady:
  - Pętla `for` - brak warunku (wyrażenia sterującego) w instrukcji **`for`** oznacza, że zawsze jest on prawdziwy:

    ```
    for (;;)
       cout << "ta pętla jest nieskończona" << endl;
    ```

  - Pętla `while` - wartość `true` lub stała 1 oznacza, że warunek zawsze jest prawdziwy.

    ```
    while (true)
       cout << "ta pętla jest także nieskończona" << endl;
    ```

  - Pętla `do-while` - wartość `true` lub stała 1 oznacza, że warunek zawsze jest prawdziwy

    ```
    do {
       cout << "i ta pętla także jest nieskończona" << endl;
    } while (true);
    ```

- Jeśli stosujemy pętlę nieskończoną, musimy znaleźć inny sposób zakończenia pętli, niż przez sprawdzenie warunku powtarzania czyli musimy użyć instrukcję skoku.

Instrukcje `break` i `continue`

- Dają możliwość kontrolowanego zakończenia powtarzania pętli w inny sposób niż poprzez sprawdzanie warunku powtarzania pętli.

- **`break`** - może wystąpić instrukcjach powtarzania `for`, `while`, `do-while`; powoduje przerwanie najciaśniej otaczającej ją instrukcji pętli (najbardziej zagnieżdżonej pętli) i przejście do instrukcji następnej po przerwanej pętli
- Instrukcje `break` występuje również w instrukcji `switch` . Powoduje tam zakończenie wykonywania instrukcji związanych z dopasowanym przypadkiem i przejście do instrukcji występującej po `}` zamykającym instrukcję `switch`.

- **`continue`** - może wystąpić tylko w instrukcjach powtarzania `for`, `while`, `do-while`; powoduje pominięcie wykonania instrukcji do końca najciaśniej otaczającej ją pętli i przejście do miejsca wznawiania tej pętli.

- **Przykład 1.** Pobieraj znaki z klawiatury do wprowadzenia litery a

```
for (;;) {
  znak=cin.get();
  if (znak=='a') break;
  cout << "Wpisałeś znak: " << znak << endl;
}
cout << "Koniec! Wpisałeś literę a." << endl;
```

przejdź do pierwszej instrukcji za pętlą `for`

Przykład 2. Wczytuj liczby i obliczaj ich pierwiastek kwadratowy, liczby ujemne pomijaj. Jeśli chcesz zakończyć, wprowadź 0.

```
x=-1;
while ( x ) // skrót while (x != 0)
{
  cin >> x;
  if (x<0) continue;
  if (x>0) cout << "Pierwiastek: " << sqrt(x) << endl;
}
```

wróć do sprawdzenia warunku

```
do {
  cin >> x;
  if (x<0) continue;
  if (x>0) cout << "Pierwiastek: " << sqrt(x) << endl;
} while (x);
```

wróć do sprawdzenia warunku

```
for (i=0; i<10; ++i) {
  cin >> x;
  if (x<0) continue;
  if (x>0) cout << "Pierwiastek: " << sqrt(x) << endl;
}
```

wykonanie części inkrementacyjnej, następnie sprawdzenie warunku

Przykład 3. Obliczyć średnią ocenę. Wczytuj oceny do napotkania 0.

```
#include <iostream.h>
int main() {
  int Ilosc;
  double Ocena, Suma;
  Ilosc=Suma=0;
  while (true) {
    cout << "Wpisz ocene. Konczy wpisanie l.ujemnej: ";
    cin >> Ocena;
    if (Ocena<0) break;
    ++Ilosc;
    Suma+=Ocena;
  }
  if (!Ilosc)  cout << "Brak ocen" << endl;
  else  cout << "Srednia ocena to: " << Suma/Ilosc << endl;
  return 0;
}
```

Przykład 4. Obliczanie pierwiastka kwadratowego

```
#include <iostream.h>
#include <math.h>
main () {
  double x;
  do
  {
    cout << "Podaj liczbę dodatnia: ";
    cin >> x;
    if (x < 0) {
      cout << "Prosiłem o liczbę dodatnią!" << endl;
      continue;
    }
    else if (x > 0)
      cout << "Pierwiastek kwadratowy wynosi: " << sqrt(x) << endl;
    else  {
      cout << "Koniec." << endl;
      break;
    }
  }
  while (true);
  return 0;
}
```

Instrukcja skoku `goto`

- Jest to instrukcja, która nigdy nie jest konieczna i w praktyce zawsze można się bez niej obejść.
- Powszechnie uważa się, że instrukcja ta wprowadza bałagan do programu i czyni go nieczytelnym.
- Składnia:

  `goto etykieta`

  gdzie etykieta (ang. *label*) to identyfikator z dwukropkiem, który znajduje się w tej samej funkcji, w której odwołuje się do niej `goto`.
- Przykład :

```
x=1;
powtorz:
  x++;
if (x<10) goto powtorz;
```

- Instrukcja `goto` jest stosowana do zaniechania przetwarzania w głęboko zagnieżdżonych strukturach programu, do jednoczesnego przerwania działania dwóch lub więcej pętli:

```
for (...)
  for (...) {
  ...
  if (niepowodzenie)
    goto stop;
    ...
  }
...
stop:
  cout << "Błąd w programie";
```

- Z instrukcji `goto` należy korzystać tylko w sytuacjach wyjątkowych.

Tablice

- Tablica jest zestawem obiektów (zmiennych) *tego samego typu*, do których można się odwołać za pomocą wspólnej nazwy.
- Obiekty składowe tablicy noszą nazwę elementów tablicy. Dostęp do nich jest jedynie możliwy poprzez podanie położenia elementu w tablicy. Ten rodzaj dostępu nazywa się indeksowaniem (ang. *indexing*, *subscripting*).

Tablice jednowymiarowe

- Składnia deklaracji tablicy jednowymiarowej:
  ```
  typ nazwa_tablicy[wymiar];
  ```
  gdzie:
  typ tablicy określa typ wszystkich jej elementów.
  wymiar (ang. *dimension*) tablicy określa liczbę elementów, które można zapisać w tablicy.

- Przykład: deklaracja tablicy dziesięciu obiektów typu `int` ma postać
  ```
  int t[10];
  ```

- Wymiar musi być wyrażeniem stałym typu całkowitego, tzn. takim, który może obliczyć kompilator.
- Przykłady:
  ```
  int wyniki[25]     /*tablica przechowująca 25 liczb typu int */
  char alfabet[26];  /* tablica przechowujące 26 znaków */

  const int rozmiar_bufora=80;
  char bufor_we[rozmiar_bufora];
  ```

Dostęp do elementów tablicy

- Nazwa tablicy jest wspólną nazwą wszystkich elementów tablicy.
- Dostęp do elementów tablicy jest realizowany za pomocą indeksu, który wskazuje położenie danego elementu w tablicy.
- Indeks może być dowolnym wyrażeniem o wartości całkowitej, zawierającym zmienne i stałe całkowite.
- Pierwszy element tablicy ma indeks równy 0. Ostatni element tablicy ma indeks równy rozmiarowi tablicy minus 1.
- Przykład:
  ```
  int wyniki_testu[25]; // tablica o 25 elementach typu int
  wyniki_testu[0]=3;    // pierwszy element tablicy
  wyniki_testu[24]=5;   // ostatni element tablicy
  ```
- Indeks nie musi być stałą, może być obliczany. *Niebezpieczeństwo*: w języku C i C++ nie sprawdzane jest automatycznie przekroczenie zakresu indeksów tablicy. Programista musi sam zadbać o to, aby użyć właściwej wartości indeksu: *nieujemnego* i *mniejszego od rozmiaru tablicy*.

Rozmieszczenie tablicy w pamięci

- Każda tablica zajmuje *spójny* obszar pamięci. Przykład: tablica `int a[3]`, (zakładamy, że typ `int` zajmuje 2 bajty):

element	a[0]	a[1]	a[2]	a[3]
wartość	5	4	8	7
adres	1000	1002	1004	1006

- Całkowity rozmiar tablicy w bajtach można obliczyć za pomocą wzoru:
  ```
  razem_bajtow = sizeof(typ) * liczba_elementów_tablicy
  razem_bajtów = sizeof nazwa_tablicy
  ```

Nadawanie wartości elementom tablicy

Inicjalizowanie tablicy przy deklaracji

```
int liczby[3]={1,2,3}; // inicjalizacja: podane wartości wszystkich
                       // elementów
int liczby[3]={1};     // niepełna inicjalizacja:
                       // wartość pozostałych elementów równa 0
int liczby[]={1,2,3};  // automatyczne nadawanie rozmiaru:
                       //jeśli podane zostaną wartości wszystkich
                       // elementów, kompilator obliczy rozmiar
```

- Przykład: W tablicy `dni` przechowywane są liczby dni w 3 pierwszych miesiącach roku. Chcemy je wydrukować.

```
#include <iostream.h>
int main()
{
  int dni[] = {31,28,31};
  for (int i = 0; i < sizeof dni / sizeof (int); i++)
    cout << "Miesiac " << (i+1) << " ma " << dni[i] << " dni." << endl;
  return 0;
}
```

Przypisywanie wartości elementom tablicy

- Wartości można przypisać jedynie pojedynczym elementom tablicy.
- Przykład:

```
int wyniki_testu[20];  // deklaracja tablicy
wyniki_testu[0]=3; // przypisanie wartości elementowi tablicy
```

- Nie można jednej tablicy bezpośrednio przypisać innej tablicy:

```
int x[] = {0,1,2};
int y[3];
x=y;  // błąd
```

- Chcąc utworzyć kopię tablicy, trzeba kolejno kopiować każdy element tablicy.
- Przykład - przypisywanie wartości elementom tablicy i drukowanie:

```
#include <iostream.h>
 int main() {
   const int rozmiar=5;
   int x[rozmiar], y[rozmiar];
   int i;
   // przypisywanie wartości elementom tablicy x
   for (i=0; i<rozmiar; ++i)
     x[i]=i;
   // kopiowanie tablicy x do y
   for (i=0; i<rozmiar; ++i)
     y[i]=x[i];
   // drukowanie tablicy y
   for (i=0; i<rozmiar; ++i)
     cout << y[i] << ' ';
   cout << endl;
   return 0;
 }
```

Wczytywanie wartości elementów tablicy

- Wartości można wczytywać tylko do pojedynczych elementów tablicy.

```
int wyniki_testu[20];
for (int i=0;i<3;i++)
  cin << wyniki_testu[i];
```

Przykłady użycia tablic jednowymiarowych

- **Przykład 1.** Należy wczytać dzienne pomiary temperatury w tygodniu i obliczyć ich średnią.

```
#include <iostream.h>
 int main() {
   int const TYDZIEN=7;
   int i;
   double temperatura[TYDZIEN], suma=0 ;
   for (i=0;i<TYDZIEN;i++) {
     cin >> temperatura[i];
     suma += temperatura[i];
   }
   cout << "Srednia temperatura w tygodniu: " << suma/TYDZIEN << endl;
   return 0;
 }
```

- **Przykład 2.** Należy wczytać do 10 liczb i wydrukować je w odwrotnej kolejności.

```
#include <iostream.h>
int main()
{
  const int MAX=10;
  int t[MAX], ile; // ile - faktyczna liczba elementów
  cout << "Podaj liczbe elementow: ";
  cin >> ile;
  if (ile < 0 || ile > MAX) {
    cout << "Czy dobrze podales liczbe elementow?" << endl;
    return 1;
  }
  cout << "Podaj liczby:" << endl;
  /* wpisanie liczb do tablicy */
  for (int i=0; i<ile; i++)
    cin >> t[i];
  /* wyświetlenie liczb w tablicy */
  cout << "Liczby w odwrotnej kolejnosci:" << endl;
  for (int i=ile-1; i>=0; i--)
    cout << t[i] << ' ';
  cout << endl;
  return 0;
}
```

- **Przykład 3:** Obliczyć krotność występowania cyfr 0-9 we wprowadzonym tekście.

```
#include <iostream.h>
#include <iomanip.h>
int main() {
  int znak,i;
  int cyfry[10];
  for (i=0;i<10;++i)
    cyfry[i]=0;
  while ((znak=cin.get()) != '\n')
    if (znak>='0' && znak <='9')
      ++cyfry[znak-'0'];
  cout << "Wprowadzone cyfry" << endl;
  for (i=0;i<10;++i)
    cout << setw(3) << i;
  cout << endl;
  for (i=0;i<10;++i)
    cout << setw(3) << cyfry[i];
  return 0;
}
```

Tablice wielowymiarowe

- Deklaracja tablicy wielowymiarowej:
  typ nazwa_tablicy[*wymiar_n*]...[*wymiar_2*][*wymiar_1*]

Tablica dwuwymiarowa

- Najprostszą tablicą wielowymiarową jest tablica dwuwymiarowa.
- Deklaracja tablicy dwuwymiarowej:
  typ nazwa_tablicy[*liczba_wierszy*][*liczba_kolumn*]
- Przykład deklaracji tablicy $A_{5,10}$:

```
int A[5][10]; /* w deklaracji każdy nawias określa
                 liczbę elementów,
                 tablica ma 5 wierszy i 10 kolumn */
```

- Pierwszy i ostatni element tablicy A:

```
int A[0][0];    /* [wiersz][kolumna] */
int A[4][9];    /* [wiersz][kolumna] */
```

- Dostęp do elementu $a_{1,2}$:

```
a[1][2]=5;
```

- Elementy tablicy umieszczane są w pamięci wierszami. Przykład tablicy `int a[2][3]` (zakładamy, że typ `int` zajmuje 2 bajty), w której umieszczono kolejne liczby całkowite poczynając od 0.

Element	Adres	Zawartość
a[0][0]	1000	0
a[0][1]	1002	1
a[0][2]	1004	2
a[1][0]	1006	3
a[1][1]	1008	4
a[1][2]	1010	5

Inicjalizacja tablic dwuwymiarowych

- Inicjalizacja tablic dwuwymiarowych jest podobna do inicjalizacji tablic jednowymiarowych.
- Przykład 1: Wartości każdego wiersza umieszczamy w nawiasach klamrowych

```
int a[2][3] = { {0,1,2},
                {3,4,5}};
```

- Przykład 2: Jeśli wpisujemy wszystkie wartości, to możemy pominąć nawiasy wyznaczające wiersze, musimy tylko pamiętać, że wartości początkowe nadawane są kolejnymi wierszami.

```
int a[2][3] = { 0,1,2
                3,4,5 };
```

- Przykład 3: Można pominąć ostatni wymiar

```
int a[][3] = { 0,1,2
               3,4,5 };
```

- **Przykład 1:** Wypełnienie tablicy wartościami 1 i wyświetlenie tablicy wierszami.

```
#include <iostream.h>
#include <iomanip.h>
int main()
{
  const int MAXW=3;
  const int MAXK=4;
  int a[MAXW][MAXK];

  /* wpisanie liczb do tablicy */
  for (int i=0; i<MAXW; ++i)
    for (int j=0; j<MAXK; ++j)
      a[i][j]=1;

  /* wyświetlenie liczb w tablicy */
  for (int i=0; i<MAXW; ++i){
    for (int j=0; j<MAXK; ++j)
      cout << setw(2) << a[i][j];
    cout << endl;
  }
  return 0;
}
```

Przykład 2: Wyzerowanie głównej przekątnej w macierzy kwadratowej, przypisanie pozostałym elementom wartości 1.

```
#include <iostream.h>
#include <iomanip.h>
int main()
{
  const int MAX=4;
  int a[MAX][MAX],i,j;

  /* wpisanie liczb do tablicy */
  for (i=0; i<MAX; i++)
    for (j=0; j<MAX; j++)
      a[i][j]= (i==j) ? 0:1;

  /* wyświetlenie liczb w tablicy */
  for (i=0; i<MAX; ++i){
    for (j=0; j<MAX; ++j)
      cout << setw(3) << a[i][j];
    cout << endl;
  }
  return 0;
}
```

Przykład 3. Obliczanie i wyświetlanie sumy elementów w każdej kolumnie tablicy.

```
#include <iostream.h>
int main()
{
  const int MAXW=3;
  const int MAXK=4;
  int t[MAXW][MAXK];
  int suma;
  /* wpisanie liczb do tablicy */
  for (int i=0; i<MAXW; ++i)
  {
    cout << "Podaj " << MAXK << " elementow wiersza " << i <<endl;
    for (int j=0; j<MAXK; ++j)
      cin >> t[i][j];
  }
  /* obliczenie sum w kolumnach */
  for (int j=0; j<MAXK; ++j){
    suma =0;
    for (int i=0; i<MAXW; ++i)
      suma += t[i][j];
    cout << "Suma w kolumnie " << j << " wynosi: " << suma << endl;
  }
  return 0;
}
```

Przykład 4. Przepisanie tablicy do wektora.

```
#include <iostream.h>
#include <iomanip.h>
int main()
{
  const int MAXW=3;
  const int MAXK=4;
  int t[MAXW][MAXK];
  int w [MAXW*MAXK];
  int i,j,iw;

  /* wpisanie liczb do tablicy */
  for (i=0; i<MAXW; ++i)
  {
    cout << "Podaj " << MAXK << " elementow wiersza " << i <<endl;
    for (j=0; j<MAXK; ++j)
      cin >> t[i][j];
  }

  /* przepisanie tablicy do wektora */
  iw=0;
  for (i=0; i<MAXW; ++i)
    for (j=0; j<MAXK; ++j)
    {
      w[iw]= t[i][j];
      iw++;
    }
  /* wydrukowanie wektora */
  cout << "Elementy wektora" << endl;
  for (i=0; i<MAXW*MAXK; i++)
    cout << setw(3) << w[i];
  return 0;
}
```

Przykład 5. Zamiana dwóch wybranych wierszy tablicy

```
#include <iostream.h>
#include <iomanip.h>
int main()
{
  const int MAXW=3;
  const int MAXK=4;
  int t[MAXW][MAXK];
  int i,j,tymcz, w1, w2;

  /* wpisanie liczb do tablicy */
  for (i=0; i<MAXW; ++i)
  {
    cout << "Podaj " << MAXK << " elementow wiersza " << i <<endl;
    for (j=0; j<MAXK; ++j)
      cin >> t[i][j];
  }
  cout << "Ktore wiersze zamienic?";
  cin >> w1 >> w2;

// pominięte sprawdzenie, czy użytkownik podał numery wierszy z zakresu

  /* Zamiania wierszy */
  for (j=0; j<MAXK; j++)
    {
      tymcz=t[w1][j];
      t[w1][j]= t[w2][j];
      t[w2][j]=tymcz;
    }
  cout << "Tablica po zamianie" << endl;
  for (i=0; i<MAXW; i++)
    {
    for (j=0; j<MAXK; j++)
      cout << setw(3) << t[i][j];
    cout << endl;
    }
  return 0;
}
```