Tablice

Tablica to struktura danych, która może przechowywać wiele wartości tego samego typu. Na przykład tablica może zawierać:

- 10 wartości typu **int** opisujących liczbę studentów przyjętych na kierunek matematyka w latach 2008-2017.
- 12 wartości typu **short** opisujących liczbę dni w kolejnych miesiącach.
- 365 wartości typu **float** mówiących o wydatkach na jedzenie w ciągu kolejnych dni roku.

Każda wartość jest zapisywana w osobnym elemencie tablicy, a komputer wszystkie te elementy umieszcza w pamięci jeden za drugim.
Tablica jest przykładem typu pochodnego.

Aby utworzyć tablicę, używa się deklaracji. Deklaracja tablicy powinna zawierać trzy elementy:

- typ wartości poszczególnych elementów,
- nazwę tablicy,
- liczbę elementów tablicy.

Tablice

Deklaracja

```
short miesiace[12];
```

tworzy tablicę o nazwie `miesiace` mającą 12 elementów/komórek, z których każda zawiera wartość typu **short**.

Deklaracja tablicy

nazwaTypu nazwaTablicy[wielkośćTablicy]

Wyrażenie `wielkośćTablicy`, określające liczbę elementów, może być jedną z poniższych wielkości:

- stałą całkowitą większą od zera (np. 12),
- wartością **const**,
- wyrażeniem stałym (np. `5*sizeof(int)`), w którym wszystkie wartości są znane w chwili kompilacji.

W szczególności `wielkośćTablicy` nie może być zmienną, której wartość jest ustawiana w trakcie działania programu.

Rozmiar tablicy

Kompilator musi wiedzieć, ile miejsca ma zarezerwować na daną tablicę.
Poniższy kod jest nieprawidłowy.

```
int n;
cout << "Podaj rozmiar tablicy: ";
cin >> n;
int tab[n];          Błąd!
```

Wartość *n* nie jest znana w trakcie kompilacji. Znana jest dopiero w trakcie działania programu (gdy użytkownik wprowadzi ją z klawiatury).

Uwaga: Niektóre kompilatory posiadają rozszerzenia umożliwiające skompilowanie i poprawne działanie powyższego przykładu.
Z takich kompilatorów korzysta m.in. Dev C++.
Kompilator Microsoft Visual Studio zgłosi jednak błąd.

Warto mieć jednak świadomość, że tablice o zmiennym rozmiarze (VLA - variable length arrays) nie są częścią standardu C++11.

Rozmiar tablicy

Zaletą **tablic statycznych** (tablic o stałym rozmiarze) jest łatwość ich użycia, natomiast dużą wadą jest brak możliwości sterowania ich rozmiarem. Z programu, w którym zdefiniujemy tablicę

```
int tab[100];
```

nie skorzysta użytkownik, który zechce wprowadzić 101 liczb.

Standard C++11 umożliwia tworzenie tablic do przechowywania danych, których liczby nie jesteśmy w stanie dokładnie przewidzieć w trakcie pisania programu.
Jest to tzw. *dynamiczna alokacja tablicy*.
To użytkownik, a nie kompilator, samodzielnie przydziela potrzebny obszar pamięci na tablicę, a na koniec zwalnia go.
Aby utworzyć **tablicę dynamiczną** potrzebne są wskaźniki (o czym będzie później).

Uwaga: Inny sposób to użycie bibliotecznej klasy kontenerowej `vector`.

Tablice

- Do każdego elementu tablicy odwołujemy się z osobna. Służą do tego *indeksy* pozwalające ponumerować elementy.
- Elementy indeksujemy zawsze od zera.
- Indeks podaje się w nawiasach kwadratowych; np. `miesiace[0]` to pierwszy element tablicy `miesiace`, a `miesiace[11]` to element ostatni. Jak widać, indeks ostatniego elementu jest o jeden mniejszy od liczby elementów w tablicy.
- Wobec tego deklaracja tablicy umożliwia utworzenie wielu zmiennych w jednej deklaracji; za pomocą indeksu można później wskazywać poszczególne elementy.
- Kompilator nie sprawdza, czy użyto prawidłowych indeksów; np. kompilator nie zgłosi błędu, jeśli odwołamy się do nieistniejącego elementu `miesiace[100]`. Jednak przypisanie takiemu elementowi wartości może spowodować kłopoty, w szczególności uszkodzić dane lub kod i ewentualnie zawiesić program.

Tablice

Przykład prawidłowej deklaracji tablicy i ustawienia jej wartości:

```
int tab[2];
tab[0] = 5;
tab[1] = 3;
```

Przykład nieprawidłowego wypełnienia tablicy:

```
int tab[2];
tab[0] = 5;
tab[1] = 3;
tab[2] = 6;    Błąd!
```

- Element `tab[2]` nie istnieje.
- Kompilator nie zgłosi błędu, jednak instrukcja `tab[2] = 6;` może zniszczyć w pamięci ważną daną.

```cpp
#include <iostream>
using namespace std;
int main()
{
   int cena[3];
   cena[0] = 6;
   cena[1] = 8;
   cena[2] = 5;
   cout << "Tablica cena zawiera wartości: "
  << cena[0] << ", " << cena[1] << " i " << cena[2] << endl;
   int ile[3] = {2, 1, 2};
   int koszt = cena[0]*ile[0] + cena[1]*ile[1] + cena[2]*ile[2];
   cout << "Wydano " << koszt << " zł." << endl;
   cout << "Rozmiar tablicy ile: " << sizeof(ile) << " bajtów";
   cout << "\nRozmiar jednego elementu: " << sizeof(ile[0]);
}
```

```
Tablica cena zawiera wartości: 6, 8 i 5
Wydano 30 zł.
Rozmiar tablicy ile: 12 bajtów.
Rozmiar jednego elementu: 4
```

Tablice

- Program najpierw tworzy trójelementową tablicę `cena`; jej elementy mają indeksy 0, 1 i 2.
- Pierwszy element tablicy (tzn. element o indeksie 0) ma wartość 6, drugi (o indeksie 1) ma wartość 8, ostatni trzeci (o indeksie 2) wartość 5:

```
cena[0] = 6;
cena[1] = 8;
cena[2] = 5;
```

- Każdy element tej tablicy to liczba typu **int** ze wszystkimi jej cechami, zatem program może przypisywać elementom wartości, dodawać elementy, mnożyć je i wyświetlać ich wartości.
- Tablicę można też wypełnić wartościami w chwili jej tworzenia (inicjalizacja):

```
int ile[3] = {2, 1, 2};
```

Tablice

Tablice można inicjalizować tylko podczas definiowania tablicy; nie można tego zrobić później:

```
int A[4];
A[4] = {2, 2, 3, 2};          nieprawidłowo
```

```
int A[4] = {2, 2, 3, 2};      prawidłowo
```

Nie można przypisać jednej tablicy w całości do innej tablicy:

```
int A[4] = {2, 2, 3, 2};
int B[4] = A;                 nieprawidłowo
```

```
int A[4] = {2, 2, 3, 2};
int B[4];
B = A                         nieprawidłowo
```

Tablice

Podczas inicjalizacji tablicy można podać mniej wartości niż ma tablica:

```
int A[5] = {3, 2};
```

Wtedy dwa pierwsze elementy otrzymają wartości kolejno 3 i 2, a pozostałe kompilator wypełni zerami. Powyższa inicjalizacja jest równoważna następującej:

```
int A[5] = {3, 2, 0, 0, 0};
```

Inicjalizacja

```
int A[5] = {3, 2, 6, 0, 1, 2};
```

zakończy się błędem: na liście jest 6 liczb, a tablica ma rozmiar 5.
Jeśli w inicjalizacji tablicy nawiasy kwadratowe pozostawimy puste, kompilator policzy elementy inicjalizujące i ustawi odpowiednią wielkość tablicy:

```
int A[] = {1, 5, 2, 1, 1, 4};
```

Kompilator utworzy 6-elementową tablicę A.

Wypełnianie tablicy w pętli (1)

```
#include <iostream>
using namespace std;
int main()
{
  int tab[100], n;
  cout << "Podaj rozmiar tablicy (nie większy niż 100): ";
  cin >> n;
  //Wypełnienie tablicy wartościami z klawiatury
  cout << "Podaj elementy tablicy:" << endl;
  for (int i = 0; i <= n-1; i++)
  {
    cout << "tab[" << i << "]= ";
    cin >> tab[i];
  }

  //Wyświetlenie zawartości tablicy
  cout << "Oto elementy tablicy:" << endl;
  for (int i = 0; i < n; i++)
    cout << tab[i] << " ";
}
```

Wypełnianie tablicy w pętli (1)

Przykładowy wynik działania programu:

```
Podaj rozmiar tablicy (nie większy niż 100): 5
Podaj elementy tablicy:
tab[0]= 2
tab[1]= 3
tab[2]= 1
tab[3]= 1
tab[4]= 3
Oto elementy tablicy:
2 3 1 1 3
```

W programie utworzono tablicę 100-elementową, użytkownik nie musi jednak wykorzystać jej w całości. W tym celu wprowadza rzeczywisty rozmiar n (nie większy niż maksymalny $= 100$). Kolejne operacje wykonywane są na fragmencie tablicy wskazanym przez użytkownika, np:

```
for (int i = 0; i < n; i++)
```

zamiast

```
for (int i = 0; i < 100; i++)
```

Wypełnianie tablicy w pętli (2)

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main()
{
  int tab[100], a, b, n;
  cout << "Podaj kolejno początek i koniec przedziału: \n";
  cin >> a >> b;
  cout << "Podaj rozmiar tablicy (nie większy niż 100): ";
  cin >> n;

  //Wypełnienie tablicy liczbami losowymi z przedziału [a,b]
  srand(time(NULL));
  for (int i = 0; i <= n-1; i++)
    tab[i] = rand()%(b - a + 1) + a;

  cout << "Oto elementy tablicy:" << endl;
  for (int i = 0; i < n; i++)
    cout << tab[i] << " ";
}
```

Wypełnianie tablicy w pętli (2)

Przykładowy wynik działania programu:

```
Podaj kolejno początek i koniec przedziału:
0
9
Podaj rozmiar tablicy (nie większy niż 100): 10
Oto elementy tablicy:
1 1 9 0 7 1 7 3 0 2
```

Program zlicza jedynki w n-elementowym ciągu binarnym.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main()
{
  int bin[100], n, ile=0;
  cout << "Podaj rozmiar tablicy (nie większy niż 100): ";
  cin >> n;
  srand(time(NULL));
  for (int i = 0; i < n; i++)
    bin[i] = rand()%2;
  cout << "Oto elementy tablicy:" << endl;
  for (int i = 0; i < n; i++)
    cout << bin[i] << " ";

  for (int i = 0; i < n; i++)
    if (bin[i] == 1)
      ile++;
  cout << "\nLiczba jedynek w wylosowanym ciągu to: " << ile;
}
```

Przykładowy wynik działania programu:

```
Podaj rozmiar tablicy (nie większy niż 100): 10
Oto elementy tablicy:
1 0 1 0 0 0 1 0 0 1
Liczba jedynek w wylosowanym ciągu to: 4
```

Tablice znakowe

Definicja

```
char tab[3];
```

oznacza, że tab jest tablicą 3 elementów, będących znakami.

```
#include <iostream>
using namespace std;
int main()
{
   char tab[3];
   tab[0] = 'C';
   tab[1] = '+';
   tab[2] = '+';

   for (int i = 0; i < 3; i++)
      cout << tab[i];
}
```

```
C++
```

Tablice znakowe

W tablicy znakowej można również umieścić tekst. Wtedy po ciągu liter (a właściwie ich kodów liczbowych) następuję znak o kodzie 0 (*NULL*). Jest on po to, aby oznaczyć, gdzie kończy się ciąg liter. Na taki ciąg liter zakończony znakiem *NULL* mówimy *string* (łańcuch).

```
#include <iostream>
using namespace std;
int main()
{
  char tab1[] = {"kot ali"};
  char tab2[] = {'k', 'o', 't', ' ', 'a', 'l', 'i'};
  cout << "Rozmiar tab1: " << sizeof(tab1) << endl;
  cout << "Rozmiar tab2: " << sizeof(tab2);
}
```

```
Rozmiar tab1: 8
Rozmiar tab2: 7
```

Program sprawdza, czy wprowadzony wyraz jest palindromem.

```
#include <iostream>
using namespace std;
int main()
{
   char tab[100];
   cout << "Podaj wyraz: ";
   cin >> tab;

   int n = 0;
   while (tab[n] != NULL) n++;    //obliczenie długości wyrazu

   int i = 0, j = n-1;
   while (i < n/2)
   {
      if (tab[i] != tab[j]) break;
      i++;
      j--;
   }
   if (i < j) cout << "Wyraz nie jest palindromem.";
   else cout << "Wyraz jest palindromem.";
}
```

Przykładowe wyniki działania programu:

```
Podaj wyraz: kajak
Wyraz jest palindromem.
```

```
Podaj wyraz: owocowo
Wyraz jest palindromem.
```

```
Podaj wyraz: anna
Wyraz jest palindromem.
```

```
Podaj wyraz: programowanie
Wyraz nie jest palindromem.
```

Tablice dwuwymiarowe

Definicja

```
int tab[3][5];
```

oznacza, że `tab` jest tablicą 3-elementową, a każdy z tych elementów jest 5-elementową tablicą `int`.

- Pierwszy element `tab[0]`, to `tab[0][0]` i jest to już liczba typu `int`. Aby zatem sięgnąć po element typu `int`, potrzebujemy dwóch indeksów.
- O pierwszym indeksie można myśleć jak o wyborze wiersza, o drugim jak o wyborze kolumny.
- Tablicę `tab` można widzieć jako macierz o 3 wierszach i 5 kolumnach. Poszczególne jej elementy to:

tab[0][0]	tab[0][1]	tab[0][2]	tab[0][3]	tab[0][4]
tab[1][0]	tab[1][1]	tab[1][2]	tab[1][3]	tab[1][4]
tab[2][0]	tab[2][1]	tab[2][2]	tab[2][3]	tab[2][4]

Tablice dwuwymiarowe

Kiedy tworzymy tablicę dwuwymiarową, możemy zainicjalizować wszystkie jej elementy. Sposób inicjalizacji wynika ze sposobu inicjalizacji tablic jednowymiarowych.

Przykład inicjalizacji tablicy jednowymiarowej:

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

W przypadku tablic dwuwymiarowych każdy element sam z siebie jest tablicą, więc inicjalizujemy go tak jak to pokazano powyżej. Wobec tego inicjalizacja będzie zawierała listę inicjalizacji tablic jednowymiarowych ujętą w nawiasy i rozdzieloną przecinkami:

```
int tab[3][5] =
{
  {2, 3, 1, 0, 1},     //wartości tab[0]
  {5, 7, 3, 6, 6},     //wartości tab[1]
  {4, 1, 2, 8, 3}      //wartości tab[2]
};
```

Tablice dwuwymiarowe

Inicjalizację

```
int tab[3][5] =
{
   {2, 3, 1, 0, 1},
   {5, 7, 3, 6, 6},
   {4, 1, 2, 8, 3}
};
```

można zapisać w skróconej formie

```
int tab[3][5] = {2, 3, 1, 0, 1, 5, 7, 3, 6, 6, 4, 1, 2, 8, 3};
```

a nawet

```
int tab[][5] = {2, 3, 1, 0, 1, 5, 7, 3, 6, 6, 4, 1, 2, 8, 3};
```

Elementy tablicy wielowymiarowej umieszczane są kolejno w pamięci komputera tak, że najszybciej zmienia się najbardziej skrajny prawy indeks. Oznacza to, że tablica przechowywana jest "rzędami".

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main()
{
  int tab[10][10], n, m;
  cout << "Podaj liczbę wierszy (n<=10): ";    cin >> n;
  cout << "Podaj liczbę kolumn  (m<=10): ";    cin >> m;
  //wypełnienie macierzy liczbami losowymi 0-9
  srand(time(NULL));
  for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
      tab[i][j] = rand()%10;
  //wyświetlenie macierzy
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < m; j++)
      cout << tab[i][j] << " ";
    cout << endl;
  }
}
```

```cpp
/*************************************************/
/* Program sortuje tablicę liczb calkowitych     */
/* w porządku niemalejącym.                      */
/* Algorytm sortowania przez wstawianie.         */
/*************************************************/
#include <iostream>
using namespace std;
int main()
{
	const int max_n = 100;
	int A[max_n], n;

	// wczytanie danych
	cout << "Ile liczb chcesz posortować? (max=100) ";
	cin >> n;
	cout << "Wprowadź liczby " << endl;
	for (int i = 0; i < n; i++)
	{
		cout << "A[" << i << "] = ";
		cin >> A[i];
	}

	// wypisanie tabeli
	cout << "Tabela przed sortowaniem:" << endl;
	for (int i = 0; i < n; i++)
		cout << A[i] << "  ";

	// sortowanie przez wstawianie
	int i, j, pom;
	for (i = 0; i < n-1; i++)
	{
		j = i + 1;
		pom = A[j];
		while ((j > 0) && (A[j-1] > pom))
		{
			A[j] = A[j-1];
			j--;
		}
		A[j] = pom;
	}

	// wypisanie posortowanej tabeli
	cout << "\nTabela po sortowaniu:" << endl;
	for (i = 0; i < n; i++)
		cout << A[i] << "  ";
}
```

PRZYKŁAD DZIAŁANIA PROGRAMU:

```
Ile liczb chcesz posortować? (max=100) 6
Wprowadź liczby
A[0] = 5
A[1] = 2
A[2] = 8
A[3] = 0
A[4] = 3
A[5] = 6
Tabela przed sortowaniem:
5  2  8  0  3  6
Tabela po sortowaniu:
0  2  3  5  6  8
```

```cpp
/******************************************** */
/* Program sortuje tablicę liczb całkowitych    */
/* w porządku niemalejącym.                     */
/* Algorytm sortowania bąbelkowego.             */
/******************************************** */
#include <iostream>
using namespace std;
int main()
{
	const int max_n = 100;
	int A[max_n], n;

	// wczytanie danych
	cout << "Ile liczb chcesz posortować? (max=100) ";
	cin >> n;
	cout << "Wprowadź liczby " << endl;
	for (int i = 0; i < n; i++)
	{
		cout << "A[" << i << "] = ";
		cin >> A[i];
	}

	// wypisanie tabeli
	cout << "Tabela przed sortowaniem:" << endl;
	for (int i = 0; i < n; i++)
		cout << A[i] << "  ";

	// sortowanie bąbelkowe
	int i, j, pom;
	for (i = 1; i <= n; i++)
	{
		for (j = n-1; j >= i; j--)
		{
			if (A[j] < A[j-1])
			{
				pom = A[j-1];
				A[j-1] = A[j];
				A[j] = pom;
			}
		}
	}

	// wypisanie posortowanej tabeli
	cout << "\nTabela po sortowaniu:" << endl;
	for (i = 0; i < n; i++)
		cout << A[i] << "  ";
}
```

PRZYKŁAD DZIAŁANIA PROGRAMU:

```
Ile liczb chcesz posortować? (max=100) 6
Wprowadź liczby
A[0] = 5
A[1] = 2
A[2] = 8
A[3] = 0
A[4] = 3
A[5] = 6
Tabela przed sortowaniem:
5  2  8  0  3  6
Tabela po sortowaniu:
0  2  3  5  6  8
```

```cpp
/* Program zamienia miejscami elementy macierzy leżące po przeciwnej stronie */
/* prostej pionowej dzielącej macierz na dwie równe części. */
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main ()
{
  int A[10][10], n, m;
  int i, j, pom;
  cout << "Podaj liczbę wierszy (n<=10): ";
  cin >> n;
  cout << "Podaj liczbę kolumn  (m<=10): ";
  cin >> m;
  //wypełnienie macierzy liczbami losowymi 0-9
  srand(time(NULL));
  for (int i = 0; i < n; i++)
  {
       for (int j = 0; j < m; j++)
       {
             A[i][j] = rand()%10;
       }
  }
  //wyświetlenie tablicy
  cout << "Wylosowana tablica:" << endl;
  for (int i = 0; i < n; i++)
  {
       for (int j = 0; j < m; j++)
       {
             cout << A[i][j] << " ";
       }
       cout << endl;
  }
  //zamiana elementów w tablicy
  for (i = 0; i < n; i++)
  {
       for (j = 0; j < m/2; j++)
       {
             pom = A[i][j];
             A[i][j] = A[i][(m-1)-j];
             A[i][(m-1)-j] = pom;
       }
  }
  //wyświetlenie tablicy zmodyfikowanej
  cout << "Tablica zmodyfikowana:" << endl;
  for (int i = 0; i < n; i++)
  {
       for (int j = 0; j < m; j++)
       {
             cout << A[i][j] << " ";
       }
       cout << endl;
  }
}
```

PRZYKŁAD DZIAŁANIA PROGRAMU:

```
Podaj liczbę wierszy (n<=10): 3
Podaj liczbę kolumn  (m<=10): 7
Wylosowana tablica:
8 4 2 5 6 5 2
4 6 4 2 2 3 4
8 8 7 0 9 3 9
Tablica zmodyfikowana:
2 5 6 5 2 4 8
4 3 2 2 4 6 4
9 3 9 0 7 8 8
```