1. Tablice (wielowymiarowe)

Przypomnienie tablic jednowymiarowych

Na poprzednich zajęciach zajmowaliśmy się tablicami **jednowymiarowymi**, tworzonymi **statycznie**.
typ_zmiennej nazwa_zmiennej [liczba_elementow];

Tworząc tablicę statyczną musimy jednak pamiętać, że liczba elementów tablicy **musi być znana od początku**. Oznacza to, że użytkownik nie może określać rozmiaru tablicy w trakcie działania aplikacji (wspomniane ograniczenie dotyczy tylko i wyłącznie tablicy tworzonej w powyższy sposób). **Nie dotyczy to tablic tworzonych dynamicznie, ten temat jednak poruszymy po wskaźnikach.**

Przykład: int tablicaLiczb[5] = {1, 2, 5, 7, 9};

Oprócz tablic jednowymiarowych, są również **tablice tablic**, czyli **tablice wielowymiarowe. Tablicę dwuwymiarową** możemy sobie wyobrazić jako arkusz excel, macierz lub tablicę składającą się z określonej liczby wierszy i kolumn. **Indeksowanie również zaczyna się od zera.** Przypisanie i pobranie wartości komórki wymaga podania jej „współrzędnych" – obu wartości.

Inicjacja tablicy polega na podaniu ilości wierszy i kolumn:
typ_elementów_tablicy nazwa_tablicy [ilość wierszy][ilość kolumn];

Stworzenie dwuwymiarowej tablicy liczb int, w sumie 25 komórek to:
int tab [5][5];

Wartości początkowe możemy nadać przy deklaracji (tak zwana lista inicjalizacyjna):
int tab[3][2] = {{1,3},{4,5},{0,1}};
int tab[3][4] = {{2,4},{3,7,8}};
int tab[3][4] = {{2,4,0,0},{3,7,8,0},{0,0,0,0}};

Jak w przypadku tablic jednowymiarowych, nie wpisane wartości zostaną **uzupełnione zerami** (druga i trzecia tablica to to samo).

W inicjalizowanej tablicy wielowymiarowej można pominąć jej **pierwszy rozmiar**. Poniższe dwie deklaracje dają taki sam efekt:
int tab[][4] = {{1,2,3,4},{5,6,7,8}};
int tab[2][4] = {{1,2,3,4},{5,6,7,8}};

Do **odczytu i zapisu** danych w tablicach dwuwymiarowych używa się zagnieżdżonych pętli iteracyjnych FOR.

Zadeklarowana tablica - int tab [3] [4] - będzie miała indeksowanie:

X [0][0]	X[0][1]	X[0][2]	X[0][3]
X[1][0]	X[1][1]	X[1][2]	X[1][3]
X[2][0]	X[2][1]	X[2][2]	X[2][3]

Przykład 1.

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

int main() {

 //Zapisywanie
     int tab[3][4]; //tworzymy tablicę typu int o wymiarach 3x4

     for(int i = 0; i<3; i++) //przejście do wierszy (i)
     {
           for(int j = 0; j<4; j++) //przebieg po kolumnach (j)
           {
            cout<<"Podaj liczby: ";
            cin >> tab[i][j];
           }
     }

 //Odczytywanie
     for(int i = 0; i<3; i++)
     {
           for(int j = 0; j<4; j++)
           {
            cout << "tab[" << i << "][" << j << "]: " << tab[i][j] <<endl;
           }
     }
}
```

Przypadku tablic trójwymiarowych [ ] [ ] [ ], użyjemy trzech zagnieżdżonych pętli FOR, a w przypadku tablic czterowymiarowych czterech (takich i wyższych praktycznie się nie używa w programowaniu).

Przykład 2.

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

int main() {

 //Zapisywanie
 int tab[2][3] = {}; //tworzymy tablicę typu int o wymiarach 2 x 3

 //Odczytywanie tablicy bez wartości (powinny być 0)
     for(int i = 0; i<2; i++)
     {
           for(int j = 0; j<3; j++)
           {
            cout << "tab[" << i << "][" << j << "]: " << tab[i][j] <<endl;
           }
     }

cout<<"============================="<<endl;
//wpisywanie ręczne w niektóre indeksy:
cout<<"Podaj [0][0] - pierwszy wiersz (0), pierwsza kolumna (0): ";
cin>>tab[0][0];
cout<<"Podaj [0][1] - pierwszy wiersz (0), druga kolumna (1): ";
cin>>tab[0][1];
cout<<"Podaj [1][2] - drugi wiersz (1), trzecia kolumna (2): ";
cin>>tab[1][2];
cout<<"============================="<<endl;

 //Odczytywanie tablicy
     for(int i = 0; i<2; i++)
     {
           for(int j = 0; j<3; j++)
           {
            cout << "tab[" << i << "][" << j << "]: " << tab[i][j] <<endl;
           }
     }
}
```

Przykład 3.

```
#include<iostream>
using namespace std;
int main()
{

  int tab[4][5];

  for(int i=0; i<4; i++)
  {
    cout<<"{";
    for(int j=0; j<5; j++)
    {
      //wygenerowanie liczb z zakresu 1 - 10
      tab[i][j] = (( rand() % 10 ) + 1 );
      //wyświetlenie wylosowanej liczby
      cout<<tab[i][j]<<", ";
    }
    cout<<"}";
    cout<<endl;
  }
}
```

Losowanie liczb z określonego zakresu

```
int wylosowana_liczba = ( rand() % ile_liczb_w_przedziale ) + startowa_liczba;
```

Przykład 4.

```
#include<iostream>
using namespace std;
int main()
{

 char tyg2[7][20] ={
       {'P', 'o', 'n', 'i', 'e', 'd', 'z', 'i', 'a', 'l', 'e', 'k'},
       {'W', 't', 'o', 'r', 'e', 'k'},
       {'S', 'r', 'o', 'd', 'a'},
       {'C', 'z', 'w', 'a', 'r', 't', 'e', 'k'},
       {'P', 'i', 'a', 't', 'e', 'k'},
       {'S', 'o', 'b', 'o', 't', 'a'},
       {'N', 'i', 'e', 'd', 'z', 'i', 'e', 'l', 'a'}
   };

   for (int i = 0; i < 7; ++i) {
       for (int j = 0; j < 20; ++j)
        {
           if (tyg2[i][j] != 0)
           {
              cout << tyg2[i][j];
           } else {
              cout << '0';
           }
        }
      cout <<endl;
   }

}
```

2. Wskaźniki

Wskaźnik to zmienna przechowująca **adres innej zmiennej**. Inaczej mówiąc, jest to **zmienna wskazująca na adres innej zmiennej**. Skoro jest zmienną, więc posiada także swój adres. Wskaźnik może wskazywać na zmienną, strukturę, tablicę a nawet funkcję.

Podstawowe operowanie wskaźnikami:

***** – operator wyłuskania wartości zmiennej, na którą wskazuje wskaźnik (wyciąga „wartość" ze wskaźnika)

& – operator pobrania adresu danej zmiennej, tablicy, struktury, etc. (pobiera adres zmiennej)

Wskaźnik zapisujemy podobnie jak zwykłe zmienne pamiętając tylko, aby przed nazwą **wpisać operator "*". Operator ten służy także do wyłuskania wartości ze zmiennej wskaźnikowej** (wyciągnięcia wartości ze zmiennej, na którą wskazuje).

Przykład 1.

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

int main()
{
  int numer = 12345; //zmienna liczbowa
  int *wsk = &numer; //wskaźnik wsk zawiera adres zmiennej numer

  cout<< numer<<endl;
  cout << *wsk << endl; //wyświetlenie wyłuskanej wartości wskaźnika (12345)
  cout << wsk << endl; //wyświetlenie adresu zmiennej numer
  cout << &wsk << endl; //wyświetlenie adresu samego wskaźnika
  cout << &numer << endl; //wyświetlenie adresu zmiennej numer (operator pobrania
adresu)
}
```

Wartość wskaźnika (czyli wartość zmiennej na którą wskazuje) **możemy modyfikować bez użycia nazwy zmiennej**.

Przykład 2.

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

int main()
{
  int numer = 123; //zmienna liczbowa
  int *wsk = &numer; //wskaźnik wsk zawiera adres zmiennej numer

  cout << *wsk << endl;
  cout<< numer <<endl;

  *wsk = 902; //modyfikacja wartości zmiennej przy użyciu wskaźnika
  cout << *wsk << endl;
  cout<< numer <<endl;
}
```

Przykład 3.

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

int main()
{
    int liczba = 1;
    int *wskaznikNaLiczbe; //deklaracja wskaźnika na int
    wskaznikNaLiczbe = &liczba; //przypisanie wskaźnikowi adresu int

    cout<< "==============="<<endl;
    cout << "Zmienna liczba = " << liczba
    << "\nwskaznikNaLiczbe - wartość zmiennej = "
    << *wskaznikNaLiczbe << endl;

    //dwa sposoby wyświetlenia adresu zmiennej ========
    cout<< "==============="<<endl;
    cout << "Adres liczby = " << &liczba
    << "\nwskaznikNaLiczbe - adres liczby = "
    << wskaznikNaLiczbe << endl;

    //zmiana wartości za pomocą wskaźnika
    cout<< "==============="<<endl;
    *wskaznikNaLiczbe = *wskaznikNaLiczbe + 1;
    cout << "Liczba = " << liczba;
}
```

Przykład 4.

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

int main()
{
    int liczba = 12;
    string napis = "To jest napis:)";

    cout << "Wartość zmiennej liczba wynosi " << liczba << endl;
    cout << "Wartość zmiennej napis = " << napis << endl;

    cout << "Wielkość zmiennej liczba wynosi " << sizeof(liczba) << endl;
    cout << "Wielkość zmiennej napis wynosi " << sizeof(napis) << endl;

    cout << "Adres zmiennej liczba to " << &liczba << endl;
    cout << "Adres zmiennej napis to " << &napis << endl;
}
```

Przykład 5.

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

int main()
{
    int liczba = 12;
    int *wsk_liczba;

    wsk_liczba = &liczba;   // przypisanie wskaźnikowi adresu zmiennej liczba

    // pokazanie wartości zmiennej liczba na dwa sposoby
    cout << "Pokazanie wartości zmiennej liczba na dwa sposoby" << endl;
    cout << "Wartość zmiennej liczba wynosi (liczba) " << liczba << endl;
    cout << "Wartość zmiennej liczba wynosi (*wsk_liczba) " << *wsk_liczba << endl
<< endl;

    // pokazanie adresu zmiennej liczba na dwa sposoby
    cout << "Pokazanie adresu zmiennej liczba na dwa sposoby" << endl;
    cout << "Adres zmiennej liczba to (&liczba) " << &liczba << endl;
    cout << "Adres zmiennej liczba to (wsk_liczba) " <<  wsk_liczba << endl << endl;

    // wykorzystanie wskaźnika do zmiany wartości zmiennej liczba
    cout << "Za pomocą wskaźnika można zmienić wartość zmiennej liczba:" << endl;
    *wsk_liczba += 1; //Zmiana wartości przy użyciu wskaźnika
    cout << "Wartość zmiennej liczba wynosi teraz " << liczba;
}
```

3. Wskaźniki – dynamiczna alokacja pamięci

Tworząc aplikację często nie wiemy jak dużo obiektów będziemy potrzebowali utworzyć. Wskaźniki pozwalają nam na dostęp do dowolnego obszaru pamięci, a więc możemy ją zarezerwować, a jej adres przekazać do procesu, który zgłosił żądanie przydziału pamięci. **Obiekty tak tworzone nazywamy tworzonymi dynamicznie.**

Dostęp do takich obiektów mamy tylko za pomocą **wskaźników i referencji** (o tym za chwilę). Ważne aby **usuwać je, gdy nie są potrzebne** (inaczej istnieje ryzyko *memory leak*).

Przykład 1.

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

int main()
{
  int *wsk = new int;
  *wsk = 23;

  cout << "Wartość " << *wsk << endl;
  cout << "Adres " << wsk << endl;

  delete wsk;

  cout << "Wartość " << *wsk << endl;
  cout << "Adres " << wsk << endl;

  int liczba = 1;
  wsk = &liczba; //przypisanie wskaźnikowi nowego adresu (na zmienną liczba)
  cout << "Wartość " << *wsk << endl;
  cout << "Adres " << wsk << endl;

}
```

W powyższym przykładzie **operator delete** zwolnił pamięć przydzieloną operatorem **new**, jednak **adres nie uległ zmianie**. Oznacza to, że wskaźnik dalej wskazuje na to samo miejsce w pamięci, ale pamięć ta nie należy już do programu. W ostatnim przypadku ręcznie przekierowaliśmy adres.

4. Wskaźniki – tablica dynamiczna jedno i wielowymiarowa

Tworząc tablicę dynamiczną należy zadeklarować wskaźnik tego samego typu co jej elementy. Rezerwujemy miejsce w pamięci o określonym typie (takim jak wskaźnik). Używamy operatora **new**.

Tablicę dynamiczną używamy tak samo jak zwykłą statyczną, wskaźniki są potrzebne jedynie przy deklaracji.

Przykład 1.

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

int main()
{
    int rozmiar;
    cout << "Podaj rozmiar tablicy: ";
    cin >> rozmiar;

    int *tab = new int[rozmiar]; //dynamicznie utworzona tablica z wielkością podaną
przez użytkownika

    for (int i = 0; i < rozmiar; i++)
    {
        tab[i] = i + 1;
        cout << tab[i] << endl;
    }
    delete [] tab;
}
```

Przykład 2.

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

int main()
{
int tabSize = 0;
int *tab = 0; // wskaźnik tablicy dynamicznej

cout<<"Podaj rozmiar tablicy: ";
cin>> tabSize;  // wczytywanie rozmiaru tablicy

if(tabSize > 0 && tabSize < 100)
  {
      tab = new int[tabSize]; // deklarowanie pamięci

      for(int i = 0; i < tabSize; i++)
      { // w pętli
          tab[i] = i + 1; // wpisuję kolejne numery do tablicy
          cout<<"tab["<<i<<"]="<<tab[i]<<" "; // wypisywanie tablicy na ekranie
      }

      delete[] tab; // zwalnianie pamięci
  }
}
```

Generowanie tablicy dwuwymiarowej dynamicznej odbywa podobnie jak jednowymiarowej. Zamiast tworzyć wskaźnik do jednego wymiaru, tworzymy **tablicę wskaźników wskazujących na wymiary**.

Przykład 3.

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

int main()
{
  int ** tablica = new int * [3];

  tablica[0] = new int [3];   // wskaźnik tablica[0] wskazuje na nową tablicę
  tablica[1] = new int [3];   // wskaźnik tablica[1] wskazuje na nową tablicę
  tablica[2] = new int [3];   // wskaźnik tablica[2] wskazuje na nową tablicę

  tablica[2][2] = 999;
  cout << tablica[2][2];

  // zwalniamy pamiec
  delete [] tablica[0];
  delete [] tablica[1];
  delete [] tablica[2];
  delete [] tablica;
}
```

Przykład 4.

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

int main()
{
int wymiar = 3; //rozmiar
// tablica na wskazniki
int ** tab = new int * [wymiar];

// generowanie poszczegolnych wymiarów
  for (int i = 0; i<wymiar; i++)
      tab[i] = new int [wymiar];

// zwalnianie pamieci
  for (int i = 0; i<wymiar; i++)
      delete [] tab[i];

delete [] tab;
}
```

5. Wskaźniki – różne

Nazwa tablicy jest **wskaźnikiem na jej pierwszy element**. Oznacza to, że możemy odnieść się to tego elementu za pomocą nazwy tablicy i indeksu o wartości 0 lub poprzez ten wskaźnik.

Przykład 1.

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

int main()
{
  int tab[5] = {1 ,2, 3 , 4 ,5}; //inicjacja tablicy
  //odwołanie się za pomocą indeksu
  cout<<"Pierwszy element tej tablicy to: "<<tab[0]<<endl;
  //odwołanie się za pomocą wskaźnika
  cout<<"to także pierwszy element tej tablicy: "<<*tab;
}
```

Przykład 2.

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

int main()
{
/*Przypisując do wskaźnika adres tablicy ustawiamy wskaźnik na jej początku*/
int *wsk;
int tablica[10] = {1,2,33,4,5};
wsk = &tablica[0];
cout << *wsk <<endl;
}
```

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

int main()
{
    //tablice deklaracia inicjalizacja
    double waga[ 5 ] = { 55.3, 747.8, 1001.2, 5.2, 6.4 };
    //wskaźnik
    double *wskWaga = waga; //nazwa tabeli = adres

    //Wyświetlanie adresu i wartości wskaźnika wskWaga
    cout << "wskWaga = " << wskWaga
    << ", *wskWaga = " << *wskWaga << endl
    << "Dodawanie wskWaga + 1 ";
    wskWaga = wskWaga + 1;
    cout << "\nTeraz wskWaga = " << wskWaga
    << ", *wskWaga = " << *wskWaga << endl
    << endl;

    //Wyświetlanie zapisu tablicowego
    cout << "\nPodobienstwa tablic i wskaznikow\n"
    << "Pierwszy element tab waga[0] = "
    << waga[0] << endl;

    //Wyświetlanie zapisu wskaźnikowego
    cout << "Pierwszy element tab waga "
    "z uzyciem wskaznika *waga = "
    << * waga << endl
    << "Drugi element tab waga "
    "z uzyciem wskaznika *(waga + 1) = "
    << *(waga + 1) << endl << endl << endl;

    //porównanie wielkości tablic i wskaźników
    cout << "Tablica waga wazy " << sizeof( waga )
    << " bajtow!" << endl
    << "Jednak wskaznik na ta tablice *wsk_waga "
    << "wazy tylko " << sizeof( wskWaga )
    << " bajtów!" << endl;
}
```

Przykład 4.

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

int main()
{
// sposób poruszania się po elementach tablicy z wykorzystaniem wskaźników
  int tab[10] = {1, 2 ,4, 5, 6, 4, 3, 2, 1, 11};

  cout<<"1 element: "<<*tab<<endl; //wypisanie 1 elementu tablicy
  cout<<"4 element: "<<*(tab+3)<<endl; //wypisanie 4 elementu tablicy
  cout<<"6 element: "<<*(tab+5)<<endl; //wypisanie 6 elementu tablicy

  for(int i=0;i<10;i++)
  {
    cout<<*(tab+i)<<" "; //wypisanie elementów tablicy poruszając się po niej
wskaźnikiem
  }
}
```

Przykład 5.

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

int main()
{
 int a, *p1, *p2;

  // wskaźnik p1 ustawiamy na adres zmiennej a
  p1 = &a;
  // adres ten kopiujemy do wskaźnika p2
  p2 = p1;
  // w zmiennej a umieszczamy jakąś liczbę poprzez wskaźnik p1
  *p1 = 25;

  // wyświetlamy zawartość zmiennej a
  //poprzez wskaźnik p2 i p1

  cout << "a (*p2) = " << * p2 << endl;
  cout << "a (*p1) = " << * p1 << endl;
}
```

Przykład 6.

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

int main()
{
int tab[] = {5, 7, 9, 11, 2, 3, 6, 0};
int * p;
// ustawiamy p na adres / pierwszy element tablicy
p = &tab[0];
// przechodzimy przez kolejne elementy, aż do elementu 0
while(*p)
{
  cout << *(p++) << " ";
}
cout << endl;
// cofamy się aż do elementu 5 (początek)
do
{
  cout << * (--p) << " ";
} while(*p != 5);
cout << endl;
}
```

Przykład 7.

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

int main()
{
double x, y; //zmienne
double *min; //wskaźnik
double *max; //wskaźnik

cout << "Podaj dwie liczby: ";
cin  >> x >> y;

if (x < y)
  {
    min = &x; max = &y; //jeśli x jest mniejsze od y, zapisz w do pmin adres na x, a
do pmax adres  y
  } else //w przeciwnym razie odwrotnie
      {
        min = &y; max = &x;
      }

cout << "Min = " << *min << endl;
cout << "Max = " << *max << endl;}
```