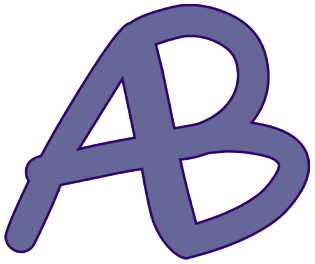


Wykład 12:

Wskaźniki
Wskaźniki i tablice
Zmienne dynamiczne

Wskaźniki

Pojęcie wskaźnika

Wskaźnik na zmienną danego typu to zmienna, która przechowuje adres zmiennej danego typu.

- ✓ Zmienne statyczne są niczym innym jak tylko obszarami pamięci operacyjnej RAM przyznanymi do przechowywania danych.
- ✓ Odwołując się do zmiennej poprzez jej nazwę odwołujemy się do przydzielonej jej pamięci.
- ✓ Wartością wskaźnika jest adres pamięci RAM, gdzie znajduje się taka zmienna.
- ✓ Adres zmiennej przechowujemy w zmiennej wskaźnikowej.

Definiowanie wskaźników

Zmienne wskaźnikowe dzielą się na różne typy – przeznaczone do przechowywania adresów różnych typów danych.

typ_wskazywanego_obiektu * nazwa wskaźnika;

Np.:

```
int *wsk_na_int;
char * wsk_na_znak;
float  * wsk_na_float;
```

Pojęcie wskaźnika

Aby uzyskać adres zmiennej statycznej, który można przechowywać w zmiennej wskaźnikowej posłużyć się można operatorem &

```
int *wskaznik;
int zmienna = 10;
```

Zdefiniowanie wskaźnika na int oraz zmiennej typu int

Przekazanie adresu „zmiennej” do wskaźnika

```
wskaznik = &zmienna;
```

Posługiwanie się wskaźnikami

`int *x;` - definicja wskaźnika do obiektów typu int

`int st;` - definicja obiektu typu int z liczbą 100

`x = &st;` - ustawienie wskaźnika na obiekt st

`cout << *x;` - wypisanie wartości obiektu wskazywanego przez x

`cin >> *x;` - zapisanie wartości do wskaźnika

```
int *x;          // definicja wskaźnika
                 // do obiektów typu int

int st = 100; // definicja obiektu typu int
                 // zainicjalizowanego liczbą 100

x = &st;         // ustawienie wskaźnika na obiekt st

cout << *x;      // wypisanie wartości obiektu
                 //  wskazywanego przez x

cin >> *x;       // to samo, co cin >> st;
```

Posługiwanie się wskaźnikami

```
 1    int *x;          // definicja wskaźnika
 2                     // do obiektów typu int
 3
 4    int st = 100; // definicja obiektu typu int
 5                     // zainicjalizowanego liczbą 100
 6
 7    x = &st;         // ustawienie wskaźnika na obiekt st
 8
 9    cout << *x;      // wypisanie wartości obiektu
10                     //  wskazywanego przez x
11
12    cin >> *x;       // to samo, co cin >> st;
```

Posługiwanie się wskaźnikami

Wskaźniki jako argumenty funkcji - przekazując wskaźniki jako argumenty funkcji sprawiamy, że z wnętrza funkcji mamy pełny dostęp do zmiennych przekazanych jako argumenty (możemy je modyfikować).
Efekt jest podobny jak przy przekazywaniu argumentów przez referencję.

```
1  #include<iostream>
2  using namespace std;
3  void zamien(int *x, int *y)
4  {
5      int pom = *x;
6          *x = *y;
7          *y = pom;
8  }
9  int main()
10 {
11         int a, b;
12         int *p1=&a,*p2=&b;
13         cin>>a>>b;
14         zamien(p1,p2); //przekazujemy adresy zmiennych
15         cout<<a<<" "<<b; //wartości zmiennych zostały zamienione
16         return 0;
17 }
```

Obsługa tablicy za pomocą wskaźników

Adres tablicy

NAZWA TABLICY

jest równocześnie

ADRESEM JEJ ZEROWEGO ELEMENTU

Dla: `int tab[10];`

zapis: `tab`

jest równoznaczny z: `&tab[0]`

Wskaźnik do tablicy

Obsługa tablic za pomocą zmiennych:

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       int tablica[]={1,2,3,4,5,6,7,8,9,10};
8       int *t;
9       t = tablica;
10      cout << "Element o indeksie 0 = " << tablica[0]<< endl;
11      cout << "Element o indeksie 0 = " << *t<< endl;
12      return 0;
13  }
```

Wskaźnik do tablicy

Po elementach tablicy możemy poruszać się na dwa sposoby:

1. przy użyciu indeksów podając kolejne numery komórek począwszy od 0,
2. przeskakując kolejne komórki tablicy z wykorzystaniem

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int tablica[]={1,2,3,4,5,6,7,8,9,10};
8      int *t;
9      t = tablica;
10     t+=3  //przesuwamy wskaźnik o trzy elementy
11     cout << "Element o indeksie 3 = " << tablica[3]<< endl;
12     cout << "Element o indeksie 3 = " << *t<< endl;
13     return 0;
14 }
```

Wskaźnik do tablicy

Przykład – wypisanie elementów tablicy poruszając się po niej wskaźnikiem.

```
1   #include<iostream>
2   #include<cstdlib>
3   using namespace std;
4   int main()
5   {
6     int tab[] = {0, 1, 2 ,3, 4, 5, 6, 7, 8, 9, 10};
7     cout<<*tab<<endl; //wypisanie  elementu tablicy o indeksie 0
8     cout<<*(tab+5)<<endl; //wypisanie 6-tego elementu tablicy
9     for(int i=0;i<11;i++)
10      cout<<*(tab+i)<<" "; //wypisanie elementów tablicy
11      cout<<endl;
12    return 0;
13  }
```

```
0
5
0 1 2 3 4 5 6 7 8 9 10
```

Przekazywanie tablic do funkcji

Przekazywanie tablic do funkcji

Tablicy nie można przesłać przez wartość.

Można tak przesłać pojedyncze jej elementy, ale nie całość.

Mamy funkcję o nagłówku:

```
void funkcja   (float tab[]);
```

która spodziewa się jako argumentu: tablicy liczb typu float, Taką funkcję wywołujemy na przykład tak:

```
float tablica[4]={ 7, 8.1, 4, 4.12};
funkcja (tablica);
```

Stałe wskaźniki
i
wskaźniki na stałe

Wskaźniki i stałe

Wskaźnik na stałą – wskazuje na zmienną typu z kwantyfikatorem const (stałą).

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       double const pi=3.14;      //stała
8       double *p1;                //zwykły wskaźnik
9       double const *p2;          //wskaźnik na stała
10      p1 = &pi;       //BŁĄD - operacja zabroniona
11      p2 = &pi;       // prawidłowo
12      *p2 = 10.0;     //BŁĄD - operacja zabroniona
13                      //nie wolno zmieniać zawartości stałej
14      return 0;
15  }
```

Wskaźniki i stałe

Stały wskaźnik – wskazuje zawsze w to samo miejse pamięci (nie można go przesuwać)

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       int x =10, y=20;        //zmiene
8       int *p1;                  //zwykły wskaźnik
9       int * const p2=&x;        //stały wskaźnik
10                                //należy go zainicjalizować,
11                                //później przypisanie wartości jest niemożliwe
12      p1 = &y;       // prawidłowo
13      p2 = &y      //BŁĄD - operacja zabroniona
14
15      return 0;
16  }
```

Przykładem stałego wskaźnika jest nazwa tablicy

Wskaźniki na funkcję

Wskaźniki na funkcję

Wskaźnik na funkcję – przechowuje adres funkcji. Może być wykorzystany do jej wywołania.

Definicja:

```
void nazwaFnkcji();
     (*wskaznik)();

int nazwaFunkcji(int a,intb,double w);
int (*wskaznik)(int,int,doble,);
```

Przypisanie adresu funkcji:

```
wskaznik = nazwa funkcji;
```

```
1  #include <iostream>
2  using namespace std;
3  typedef int (*wskNaDzialanie)(int, int);
4  int dodawanie(int a, int b);
5  int mnozenie (int a, int b);
6  int dzialanie(int a, int b, wskNaDzialanie wd);
7
8  int main()
9  {
10     cout << dzialanie(10,5,dodawanie)<<endl;
11     cout << dzialanie(10,5,mnozenie)<<endl;
12     return 0;
13 }
14 int dodawanie(int a, int b)
15 {
16     return a+b;
17 }
18 int mnozenie(int a, int b)
19 {
20     return a*b;
21 }
22 int dzialanie(int a, int b, wskNaDzialanie wd)
23 {
24     return wd(a,b);
25 }
```

Zmienne dynamiczne

Zmienne dynamiczne

Wskaźniki można zastosować do dynamicznej alokacji zmiennych – czyli rezerwacji w pamięci obszarów do przechowywania zmiennych w trakcie działania programu.

Tak stworzona zmienna nie ma nazwy, lecz tylko adres.

Adres ten przechowywany jest w statycznej zmiennej wskaźnikowej (lub bardziej skomplikowanej strukturze danych takiej jak lista lub drzewo binarne).

Zmienne dynamiczne

Do dynamicznej alokacji zmiennych służy operator new

```
int *wsk;
wsk = new int;
```

Lub krócej:

```
int *wsk = new int;
```

Operator new na podstawie typu zmiennej (lub też typu i rozmiaru tablicy) przydzieli odpowiednią ilość pamięci.

Jeśli przydział pamięci powiódł się, to wartość zmiennej **„wsk”** będzie różna od zera.

Jeśli wartość wskaźnika będzie równa 0, to pamięć nie została przydzielona. Wartość 0 bardzo często jest zastępowana stałą NULL (zalecane).

Sytuacje w których pamięć nie może zostać przydzielona:

1. rozmiar bloku pamięci, który chcesz zarezerwować jest zbyt duży;
2. system nie posiada więcej zasobów pamięci i w związku z tym nie może jej przydzielić.

Zmienne dynamiczne

Do dynamicznej alokacji zmiennych służy operator new

```
int *wsk;
wsk = new int;
```

Lub krócej:

```
int *wsk = new int;
```

Do usunięcia z pamięci zmiennej dynamicznej służy operator delete

```
delete wsk;
```

W języku **C** do przydzielania i zwalniania pamięci służyły głównie **funkcje malloc()** i **free()** w **C++** zostały one zastąpione **operatorami new** i **delete**.

Zmienne dynamiczne

Za pomocą operatora delete kasuje się tylko obiekty stworzone operatorem new

Próba skasowania czegokolwiek innego jest błędem.

Uwaga: nie należy dwukrotnie kasować obiektu.

Wyjątkiem jest zastosowanie operatora delete w stosunku do wskaźnika pokazującego na adres zerowy (NULL) – ponieważ żaden obiekt nie może mieć adresu 0 - taka konstrukcja nie powoduje błędu.

Zmienne dynamiczne

Często stosowana konstrukcja zabezpieczająca przed podwójnym kasowaniem obiektów:

```
int *wskaznik = new int;
// ……..
delete wskaznik;
wskaznik = NULL;
// …….
delete wskaźnik;  // nie spowoduje bledu
```

Zmienne dynamiczne

Cechy obiektów dynamicznych.

- Obiekty utworzone dynamicznie istnieją od momentu, gdy je utworzymy operatorem **new** do momentu, gdy je skasujemy operatorem **delete.**
- Obiekt utworzony dynamicznie nie ma nazwy. Można nim operować tylko za pomocą wskaźników.
- Obiektów takich nie obowiązują zwykłe zasady o zakresie ważności (zasady mówiące w których miejscach programu są widzialne, a w których niewidzialne pomimo, że istnieją). Jeśli tylko jest w danym momencie dostępny choćby jeden wskaźnik, który na taki obiekt pokazuje, to mamy do tego obiektu dostęp.
- Obiekty dynamicznie nie są inicjalizowane zerami (po utworzeniu zawierają przypadkowe wartości).

Tablice dynamiczne

Dynamiczna alokacja tablic:

```
int *wsk;
wsk = new int [1000];
```

Lub krócej:

```
int *wsk = new int [1000];
```

Rozmiar tablicy nie musi być stałą. Wystarczy, że jego wartość będzie znana w momencie alokacji tablicy (niekoniecznie w momencie kompilacji programu)

Tablice dynamiczne

Do usunięcia z pamięci dynamicznej tablicy służy konstrukcja:

```
delete  [] wsk;
```

Jeśli przydzieliliśmy pamięć określając ilość elementów tablicy to musimy poinformować operator delete o tym, że wskaźnik wskazywał na tablicę – inaczej usuniemy tylko jej pierwszy element.

Aby to zrobić dopisujemy zaraz za operatorem
nawiasy kwadratowe [ ]

Nie podajemy w nich rozmiaru tablicy, ponieważ operator sam ustala rozmiar bloku jaki został przydzielony.

Tablice dynamiczne

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "ile lementow ma miec tablica? ";
8      int ile;
9      cin >> ile;
10
11     int *tab = new int[ile];
12     int *p=tab;
13     for (int i=0; i<ile;i++) *p++=i;
14     p=tab;
15     for (int i=0; i<ile;i++)
16         cout << tab[i]<<" "<<endl;
17     delete [] tab;
18     return 0;
19 }
```

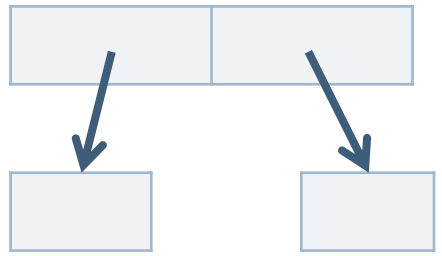
Tablice wskaźników

W tablicy przechowywać można wszystkie rodzaje zmiennych prostych, w tym także wskaźniki adresami różnych miejsc w pamięci.

`int *tabl_wsk[1000];`

```
int  *tabl_wsk[1000];
   //statyczna tablica zawierająca wskaźniki
for (int i=0; i<1000; i++)
    tabl_wsk[i] = new int;
   // tworzenie zmiennych dynamicznych
   // i zapamiętanie wskaźników do nich w tablicy
cout << *tabl_wsk[0];
   // wypisanie zmiennej dynamicznej
   // wskazywanej przez pole z tablicy
```

Tablice wskaźników do zmiennych dynamicznych



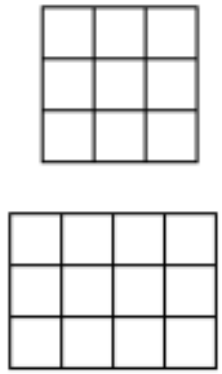

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int *tablica[10]; //STSTYCZNA tablia wskaxników
8      int temp;
9      for (int i=0; i<10;i++) //tworzymy zmienne dynamiczne
10     {                       //wskazniki do nich zapamietujemy w tablicy
11         cout<<"Tab["<<i<<"]=";
12         cin >> temp;
13         tablica[i]=new int;
14         *tablica[i]=temp;
15     }
16     for (int i=0;i<10;i++) //wypisujemy zawartość
17     {                      //zmiennych podczepionych pod tablice
18         cout << *tablica[i] <<" ";
19     }
20     for (int i=0;i<10;i++) //usuwanie elementów
21     {                      //podczepionych pod tablice
22         delete tablica[i];
23     }
24     return 0;
25 }
```

Tablica dynamiczna dwuwymiarowa

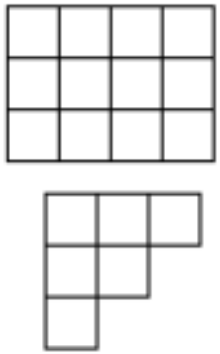
Tablica dynamiczna dwuwymiarowa to tak naprawdę tablica wskaźników do poszczególnych wymiarów (tablic).

Podczas deklaracji tablicy mamy pełną kontrolę nad wielkością poszczególnych wymiarów (tablic), statycznie nie da się osiągnąć takich efektów.

statyczna dwuwymiarowa:



dynamiczna dwywymiarowa:

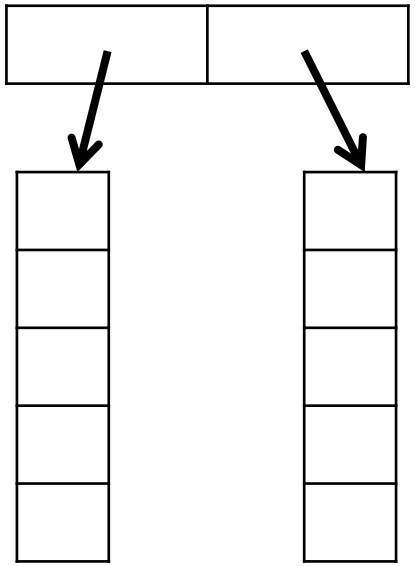


Tablica dynamiczna dwuwymiarowa

Generowanie tablicy dwuwymiarowej dynamicznej odbywa dwuetapowo.:

1. tworzymy tablicę wskaźników wskazujących na tablicę
2. tworzymy tablice jednowymiarowe, które „podczepiamy” pod tablicę główną.

int * * tablica2D



Tablica wskaźników
do tablic podrzędnych

Nazwa tablicy zapisana jest jako wskaźnik do wskaźnika do zmiennej
(takiej jak przechowywane w tablicach podrzędnych)

Tablice drugiego wymiaru nie mają nazw – tylko wskaźniki do początku zapisane w tablicy nadrzędnej

Tablica dynamiczna dwuwymiarowa

```
int ** tablica2D = new int * [2];

tablica2D[0] = new int [5];
        // wskaźnik tablica[0] wskazuje na nową tablicę
tablica2D[1] = new int [5];
        // wskaźnik tablica[1] wskazuje na nową tablicę

//posługiwanie się tablicą
tablica2D[1][2] = 100;
cout << tablica[1][2];

// usuwanie tablicy z pamięci
delete [] tablica2D[0];
delete [] tablica2D[1];
delete [] tablica2D;
```

Literatura:

W prezentacji wykorzystano przykłady i fragmenty:

- Gręboszj J. : ***Symfonia C++, Programowanie w języku C++ orientowane obiektowo***, Wydawnictwo Edition 2000.
- Jakubczyk K.: *Turbo Pascal i Borland C++ Przykłady*, Helion.

Warto zajrzeć także do:

- Sokół R. : ***Microsoft Visual Studio 2012 Programowanie w Ci C++,*** Helion.
- Kerninghan B. W., Ritchie D. M.: ***język ANSI C***, Wydawnictwo Naukowo Techniczne.

Dla bardziej zaawansowanych:

- Gręboszj J. : ***Pasja C++,*** Wydawnictwo Edition 2000.
- Meyers S.: ***język C++ bardziej efektywnie***, Wydawnictwo Naukowo Techniczne