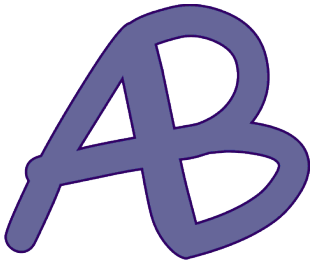


Wykład: 8

Wskaźniki

Wskaźniki

Pojęcie wskaźnika

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

Zmienne są niczym innym jak tylko komórkami pamięci operacyjnej RAM. Zatem odwołując się do zmiennej poprzez jej nazwę odwołujemy się do przydzielonej jej pamięci.
Wartością wskaźnika jest natomiast adres pamięci RAM, gdzie znajduje się taka zmienna.

Definiowanie wskaźników

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

Np.:

```
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
```

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 tablica 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 wskaźnika.

```
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:

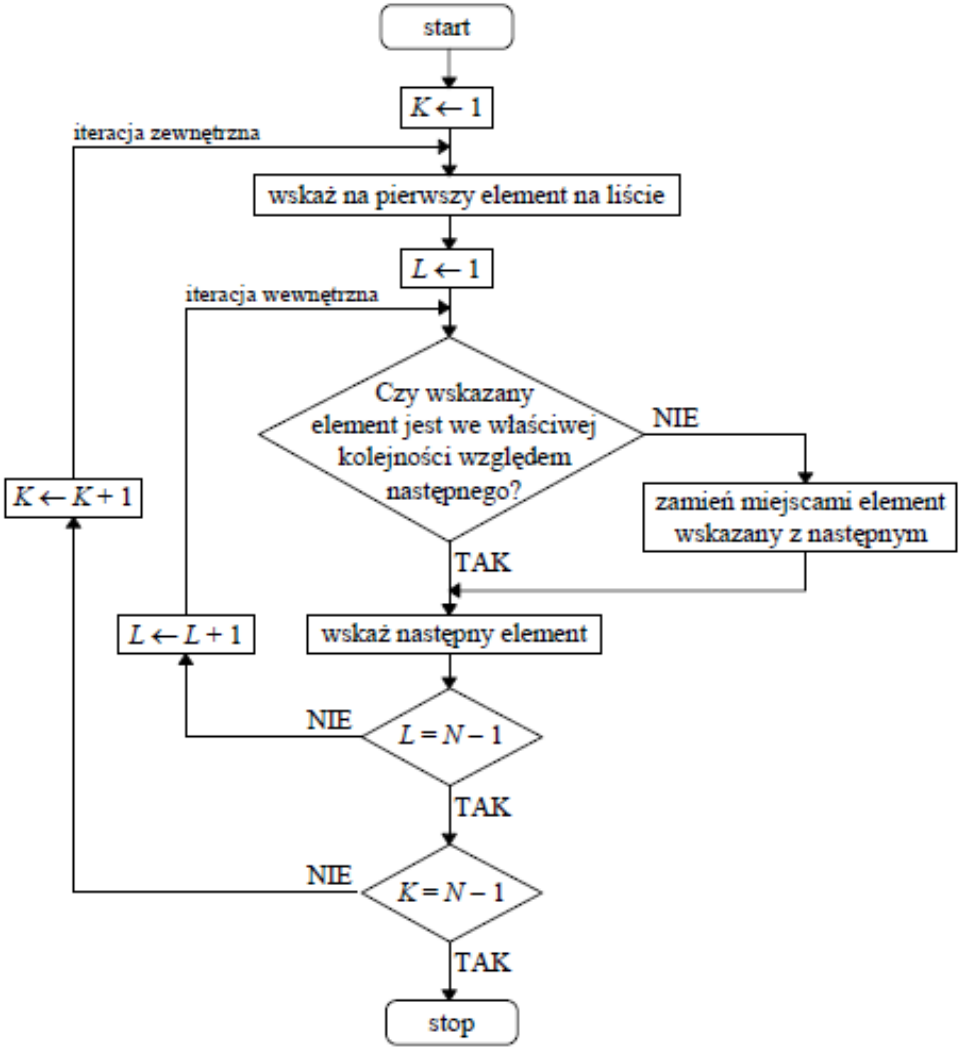
```
wskaznik = nazwa funkcji;
```

```
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 }
```

algorytmy w praktyce

- ✓ Sortowanie bąbelkowe

Przykład – sortowanie bąbelkowe



W każdym przejściu pętli wewnętrznej porównywane są ze sobą dwie kolejne wartości i w razie potrzemy są zamieniane miejscami.

W jednym cyklu pętli wewnętrznej, największa liczba (tak jak bąbelki w napoju gazowanym) w zbiorze będzie się przemieszczała na ostatnią pozycję.

W ten sposób otrzymujemy podzbiór częściowo już posortowany. Czynności te powtarzamy dla zbioru pominiętego o elementy już poukładane

Przykład – sortowanie bąbelkowe

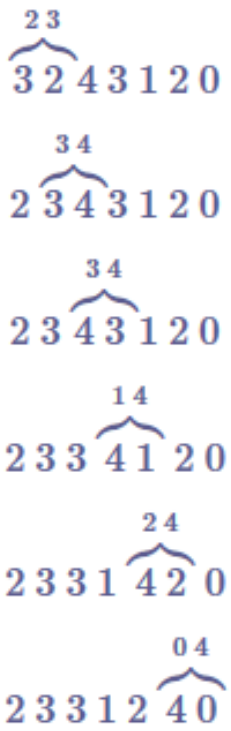
```
1   #include <iostream>
2   #include <stdlib.h>
3   #include <time.h>
4   using namespace std;
5   void losowanie(int a, int b,int t[],int n);
6   void wypisania(int t[], int n);
7   void sortowanie(int t[],int n);
8
9   int main()
10  {
11      int tab[100],n=100;
12      srand(time(NULL));
13      losowanie(-100,100,tab,n);
14      wypisania(tab,n);
15      sortowanie(tab,n);
16      wypisania(tab,n);
17      return 0;
18  }
19  void losowanie(int a, int b,int t[],int n){
23  void wypisania(int t[], int n){
27  void sortowanie(int t[],int n){
```

Przykład – sortowanie bąbelkowe

```
19  void losowanie(int a, int b,int t[],int n){
20      for (int i=0; i<n; i++)
21          t[i]=rand()%(b-a+1)+a;
22  }
23  void wypisania(int t[], int n){
24      for (int i=0; i<n; i++)
25          cout <<t[i]<<" ";
26  }
```

Przykład – sortowanie bąbelkowe

```
27  void sortowanie(int t[],int n){
28      int pom;
29      for (int j=0; j<n; j++)
30      {
31          for (int i=0;i<n-1;i++)
32              if(t[i]>t[i+1])
33              {
34                  pom=t[i];
35                  t[i]=t[i+1];
36                  t[i+1]=pom;
37              }
38          n--;
39      }
40  }
```



Literatura:

W prezentacji wykorzystano przykłady i fragmenty:

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