Języki programowania

Wskaźniki
część I

Gdzie wykorzystywane są wskaźniki?

- Praca z tablicami
- Funkcje –zmiana wartości przesyłanych argumentów
- Dostęp do specjalnych komórek pamięci
- Rezerwacja obszarów pamięci

Definiowanie wskaźników

Definiowanie wskaźników:

```
typ_obiektu *nazwa_wskaźnika;
```

nazwa_wskaźnika wskazuje na obiektu typu **typ_obiektu**.

Przykłady

Można definiować wskaźniki do obiektów różnych typów:

```
int *wi;
char *wch;
float *wf;
```

W definicji

```
int *wi;
```

wskaźnik nazywa się **wi**, i przechowuje adres pewnego obiektu typu **int**.

Wskaźnik zawiera informacje, **gdzie** znajduje się wskazywany obiekt, a nie co dany obiekt przechowuje.

Definiowanie wskaźników - uwagi

- Z definicji wskaźnika wynika, że wskaźnik pokazuje na obiekt. Referencja nie jest obiektem, dlatego nie można definiować wskaźników do referencji.

- Wskaźnik, który pokazuje na obiekt jednego typu, nie może być wykorzystany do pokazywania na obiekt innego typu.

Wskaźniki

Sama definicja wskaźnika nie powoduje, że wskaźnik wskazuje na konkretny obiekt.
Nadanie wartości obiektowi (wskaźnik wskazuje na istniejący obiekt):

```
int *wi;
int i;

w=&i; //ustawienie wskaźnika w na obiekt i
```

Jednoargumentowy operator `&` (ampersand) służy do obliczania adresu swojego argumentu stojącego po jego prawej stronie.

Jeśli wskaźnik wskazuje na konkretny obiekt, można odnosić się do tego obiektu za pomocą wskaźnika. Do operacji tej służy jednoargumentowy operator odniesienia `*`:

```
*nazwa_wskaźnika;
```

L – wartość

Możliwe są instrukcje:

```
x = *wskaznik;
*wskaznik = y;
```

Jeżeli „coś” stoi po lewej stronie operatora przypisania, to nazywa się L – wartością.

Wskaźniki – przykłady

```
int *wi; //definicja wskaźnika
int i;   //definicja zmiennej i

wi=&i; //ustawienie wskaźnika
cout<<*wi; //odniesienie do obiektu, na który wskazuje wskaźnik w
*wi=5;
```

Jaką wartość ma zmienna `i`?

```
int *wi,*wj;   //definicja wskaźnika
int i,j;       //definicja zmiennych i,j

wi=&i;
i=5;
*wi=5;
j=7;
j=*wi;
wi=&j;
wj=wi;
```

Wskaźniki typu `void`

```
void *wv;
```

Definicja wskaźnika bez podania typu obiektu, na jaki wskazuje. Może być użyty do wskazywania na obiekty dowolnego typu.

Przykłady:

```
void *wv;
int  *wi;
float *wf;

wv=wi; //teraz wskaźnik wv wskazuje na ten sam obiekt (typu int),
       //na który wskazuje wskaźnik wi
wv=wf;

wi=wf //kompilator zasygnalizuje błąd!!
wi=(int *)wf; //wykorzystanie rzutowania
wf=(float *)wv; //wykorzystanie rzutowania
```

Wskaźnik dowolnego (niestałego) typu można przypisać wskaźnikowi typu `void`. Działanie odwrotne wymaga operatora rzutowania.

```
wf = (float *) wv;
wi = (int *) wv;
```

Zastosowanie wskaźników do tablic

```
int  *wsk; //definicja wskaźnika
int tab[10]; //definicja tablicy

wsk=&tab[indeks]; //ustawienie wskaźnika na elemencie tablicy
                  //o indeksie indeks
```

```
wsk=&tab[0];
wsk=tab;
```
} instrukcje równoważne

```
wsk=&tab[indeks];
wsk=wsk+ind;
wsk += ind;
```
} przesunięcie wskaźnika o *ind* pozycji

Dodanie do wskaźnika liczby całkowitej ***ind*** powoduje, że wskaźnik pokazuje o ***ind*** elementów dalej w tablicy; niezależnie od tego, jakiego typu są elementy tablicy

$\Rightarrow$

Wskaźnik zawiera adres jakiegoś miejsca w pamięci oraz informacje, na jaki typ obiektu wskazuje

Nazwa tablicy i wskaźnik

Nazwa tablicy jest jednocześnie **adresem** jej zerowego elementu.
To stwierdzenie **nie jest** równoważne stwierdzeniu:
Nazwa tablicy jest jednocześnie wskaźnikiem do jej zerowego elementu.

```
float *wsk;
float tab[10];

wsk=tab; //wsk=&tab[0];
//możliwe jest przypisanie:
wsk++;
//niemożliwa jest instrukcja:
tab++;
tab[ind];
*(tab+ind);
```

} możliwe zastosowania

Nazwa tablicy jest **stałym wskaźnikiem** do jej zerowego elementu.

Wskaźnik jest pewnym obiektem w pamięci (to znaczy posiada swój adres).
Nazwa (również tablicy) nie jest obiektem (nie ma więc adresu).

```
float *wsk;
&wsk; //adres wskaźnika - odwołanie poprawne
```

Zastosowanie wskaźników do tablic – przykład

```
#include <iostream.h>
void main(){
        float *wf;
        float tab[10];
        wf=tab; //lub wf=&tab[0];
        for(int i=0; i<10; i++){
                *(wf++)=i/10.0;
        }
        for(i=0,wf=tab;i<10;i++,wf++)
                cout<<*wf<<endl;
}
```

Porównywanie wskaźników

Do porównania służą operatory:

== != < > <= >=

```
typ *w1, w2;
.............
if(w1 == w2){
    ...............
}

if(w1 != w2){
    ...............
}
```

Wskaźniki wskazują na ten sam obiekt

Wskaźniki wskazują na różne obiekty

Wskaźniki w argumentach funkcji

Przesyłanie argumentów przez wartość

```
float pole(float r);

void main() {
  ...
  cout << pole(7.3) << endl;
}

float pole(float r) {
  return 3.14159 * r * r;
}
```

```
float pole(float *r);
void main(){
        float rr=1;
        cout<<pole(&rr)<<endl;
        cout<<rr<<endl;
}
float pole(float *r) {
  float wynik;
  wynik= 3.14159f * *r * *r;
  *r=0.5f* *r;
  return wynik;
}
```

Przesyłanie argumentów przez wskaźnik.
Zmiana wewnątrz funkcji wartości przesyłanego argumentu.

Przesyłanie tablic do funkcji

```
#include <iostream.h>
void f1(int *pi, int rozm);
void f2(int *pi, int rozm);
void f3(int t[], int rozm);
void main(){
        int tab[5]={1,2,3,4,5};
        f1(tab,5);
        f2(tab,5);
        f3(tab,5);
}
void f1(int *pi, int rozm){
        cout<<"\nfunkcja f1\t";
        for(int i=0;i<rozm;i++)
                cout<<*(pi++)<<'\t';
}
void f2(int *pi, int rozm){
        cout<<"\nfunkcja f2\t";
        for(int i=0;i<rozm;i++)
                cout<<pi[i]<<'\t';
}
void f3(int t[], int rozm){
        cout<<"\nfunkcja f3\t";
        for(int i=0;i<rozm;i++)
                cout<<t[i]<<'\t';
}
```

Funkcje są wywoływane przez podanie nazwy tablicy

Przesłany adres tablicy inicjalizuje lokalny wskaźnik **pi**. Wewnątrz funkcji - zapis wskaźnikowy

Przesłany adres tablicy inicjalizuje lokalny wskaźnik **pi**. Wewnątrz funkcji - zapis tablicowy

Przesłany adres tablicy odebrany jako tablica

Przesyłanie tablic do funkcji – cd

- Odebranie tablicy jako tablicy – czytelność funkcji
- Odebranie tablicy jako adresu inicjującego wskaźnik – funkcja działa szybciej
- Odebranie tablicy jako adresu – łatwiejsze przekazywanie tablic wielowymiarowych (rozmiary tablicy nie muszą być znane w momencie wywołania funkcji)