Wykład nr 3

Temat: Wskaźniki i referencje.

Cytaty:

Mylić się jest rzeczą ludzką,
ale żeby coś naprawdę spaprać potrzeba komputera.

Edward Morgan Forster

Gdyby murarze budowali domy tak, jak programiści piszą programy,
to jeden dzięcioł zniszczyłby całą cywilizację.

ze zbioru prawd o programowaniu

Im bardziej zaglądał do środka, tym bardziej nic tam nie było.

A.A. Milne „Kubuś Puchatek”

Wykład nr 3

Temat: Wskaźniki i referencje.

Zakres wykładu:

- cztery domeny zastosowania wskaźników
- definiowanie wskaźników
- definiowanie referencji
- zastosowanie wskaźników wobec tablic
- przekazywanie danych do funkcji przez wskaźnik i referencję
- przekazywanie tablic do funkcji
- kwalifikator `const` w połączeniu ze wskaźnikami
- podsumowanie
- ćwiczenia powtórzeniowe i sprawdzające
- następny wykład

Cztery domeny zastosowania wskaźników

Najczęstsze zastosowania wskaźników:

1) ulepszenie pracy z tablicami (głównie chodzi o wydajność)

2) funkcje mogące zmieniać wartość przysyłanych do nich argumentów

3) dostęp do specjalnych komórek pamięci

4) dynamiczna rezerwacja obszarów pamięci

Definiowanie wskaźników

Każda zmienna ma unikalny adres wskazujący obszar pamięci zajmowany przez tą zmienną. Adres ten można przechowywać w tzw. zmiennej wskaźnikowej (wskaźniku).

Wskaźnik to obiekt (zmienna), która przechowuje adres do innego obiektu (zmiennej).

Przed użyciem wskaźnika, musimy go przypisać do konkretnego obiektu (przypisać mu adres tego obiektu). Wskaźnik można łatwo przestawić, by pokazywał na inny obiekt tego samego typu.

Do każdego wskaźnika można podstawić adres 0, zwany czasem NULL (np. `wsk=0;` lub `wsk=NULL;`). Ustawienie wskaźnika na ten adres powoduje, że wskaźnik nie pokazuje na nic sensownego.

Wskaźnik

```
float pi=3.14;   //definicja zmiennej pi
float *wsk=&pi;  //definicja wskaźnika do zmiennej typu float
```

LUB

```
float *wsk;
wsk=&pi;
```

```
cout<<"wartosc zmiennej pi: "<<pi;
cout<<"wartosc zmiennej wsk: "<<wsk;
```

→ 3.14
→ FFF0

```
cout<<"adres zmiennej pi: "<<&pi;
cout<<"adres zmiennej wsk: "<<&wsk;
```

→ FFF0
→ FFFF

```
pi=3.141;        //zmiana wartości zmiennej pi
*wsk=3.1415;     //zmiana wartości zmiennej pi za pomocą wskaźnika

cout<<"wartosc pi za pomoca wskaznika: "<<*wsk;
```

→ 3.1415

Definiowanie referencji

Referencja jest inną nazwą obiektu (zmiennej).

Podczas definiowania referencji trzeba ją od razu zainicjalizować.

Referencja nie jest kopią zmiennej, ale tą samą zmienną pod inną nazwą.

Referencja

```
int a=5;        //definicja i inicjalizacja zmiennej a
int &ref=a;     //definicja referencji o nazwie ref,
                //oraz ustawienie jej na zmienną a
```

```
cout<<"wartosc zmiennej a: "<<a;
cout<<"wartosc zmiennej ref: "<<ref;
```

→ 5
→ 5

```
ref++; //dodanie do ref wartosci 1
```

```
cout<<"wartosc zmiennej a: "<<a;
cout<<"wartosc zmiennej ref: "<<ref;
```

→ 6
→ 6

```
cout<<"adres zmiennej a: "<<&a;
cout<<"adres zmiennej ref: "<<&ref;
```

→ FFF0
→ FFF0

adres
FFF0

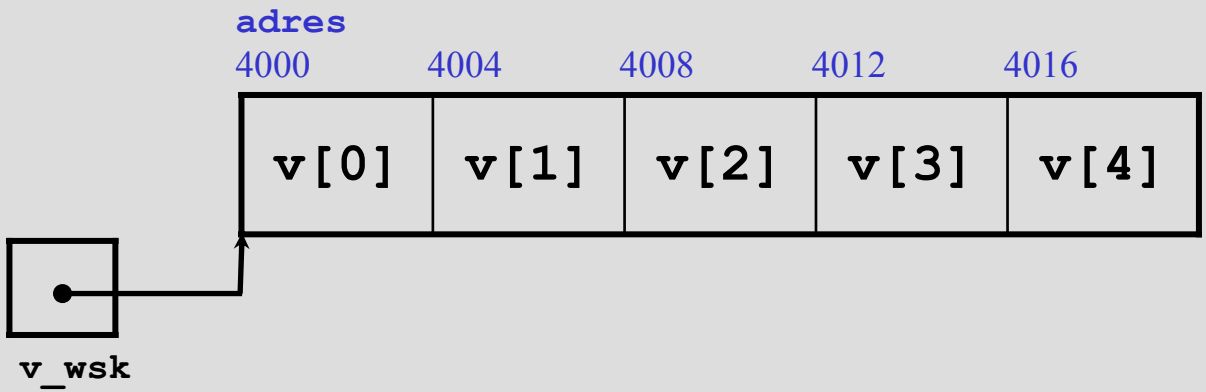
a=5
ref=5

Zastosowanie wskaźników wobec tablic

Wyrażenia wskaźnikowe i arytmetyka wskaźników:

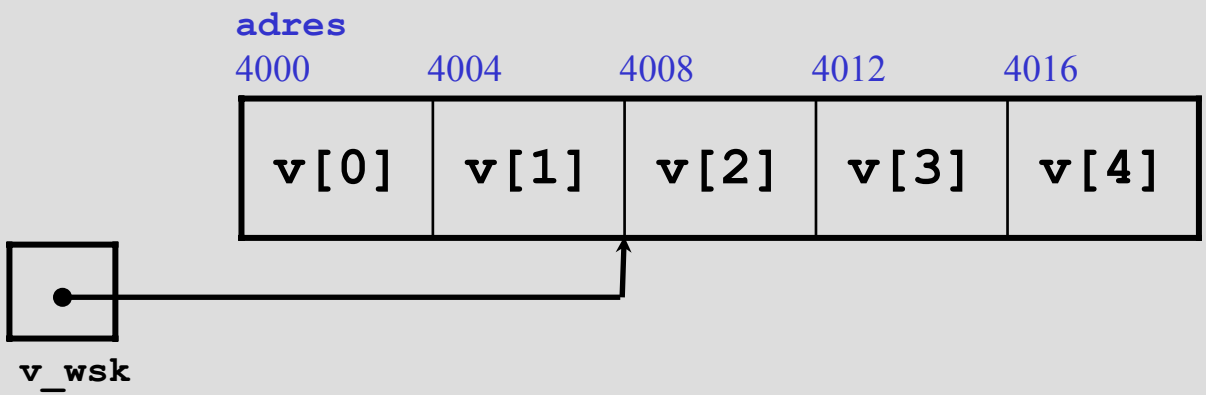
```
int v[5];                  //definicja 5-elementowej tablicy elementów typu int
int *v_wsk=v;              //definicja wskaźnika do typu int
LUB
int *v_wsk=&v[0];          //definicja wskaźnika do typu int
```

ZAPAMIĘTAJ: Nazwa tablicy jest STAŁYM wskaźnikiem do jej zerowego elementu!




```
v_wsk+=2;                  //przesunięcie wskaźnika na trzeci element tablicy
```

4008 (4000+2*4)
gdzie:
int=4 bajty



Przykład z ruchu wskaźnika wobec tablicy:

```
int *wskaznik;                                  //definicja wskaźnika do typu int
int tablica[10]={0,1,2,3,4,5,6,7,8,9};          //definicja 10-elem. tablicy typu int

wskaznik=&tablica[0];          //ustawienie wskaźnika na początek tablicy (element 0)
wskaznik=tablica;              //ustawienie wskaźnika na początek tablicy (element 0)
cout<<*wskaznik<<endl;         //wyświetlenie pierwszego elementu (liczba 0)

//tablica+=3;                  //to byłby BŁĄD!!! - nazwa tablicy jest STAŁYM WSKAŹNIKIEM!!!
wskaznik=&tablica[3];          //ustawienie wskaźnika na 4 element tablicy
cout<<*wskaznik<<endl;         //wyświetlenie 4 elementu (liczba 3)

wskaznik=wskaznik+1;           //przesunięcie wskaźnika na 5 element tablicy
cout<<wskaznik[4]<<endl;       //wyświetlenie 5 elementu (liczba 4) (notacja wskaźnik/indeks!)

wskaznik++;                    //przesunięcie wskaźnika na 6 element tablicy
cout<<*wskaznik;               //wyświetlenie 6 elementu (liczba 5)

cout<<*wskaznik++;             //wyświetlenie 6 elementu (liczba 5)
                               //i przesunięcie wskaźnika na element 7
cout<<*wskaznik<<endl;         //wyświetlenie 7 elementu (liczba 6)

cout<<*(wskaznik++);           //wyświetlenie 7 elementu (liczba 6)
                               //i przesunięcie wskaźnika na element 8
cout<<*wskaznik<<endl;         //wyświetlenie 8 elementu (liczba 7)

cout<<*(wskaznik+1);           //wyświetlenie 9 elementu (liczba 8) (bez przesuwania wskaźnika)

wskaznik+=2;                   //przesuniecie wskaźnika na 10 (ostatni element tablicy)
cout<<*wskaznik<<endl;         //wyświetlenie 10 elementu (liczba 9)

wskaznik++;                    //UWAGA: niebezpieczne, wskaźnik poza obszarem tablicy!!!
cout<<*wskaznik<<endl;         //wyświetlenie elementu spoza tablicy (śmieć)
```

Zostaną wyświetlone elementy tablicy w następującym porządku:

0 3 4 5 5 6 6 7 8 9 śmieć

Przekazywanie danych do funkcji przez wskaźnik i referencję

Przekazywanie argumentów do funkcji (odbieranie argumentów w funkcji):

– przez wartość (funkcja **pracuje na kopii** przekazywanej zmiennej, więc **NIE MA** możliwości jej modyfikowania)

– przez wskaźnik (funkcja **pracuje na oryginale** przekazywanej zmiennej, więc **MA** możliwość jej modyfikowania)

– przez referencję (funkcja **pracuje na oryginale** przekazywanej zmiennej, więc **MA** możliwość jej modyfikowania)

Domyślnie, zwykłe obiekty (zmienne) przekazywane są do funkcji przez wartość.

Ponieważ każda funkcja za pomocą instrukcji `return` zwraca tylko jedną wartość, więc funkcja otrzymująca argumenty przez wartość może zmodyfikować co najwyżej wartość jakiegoś jednego obiektu.

Chcąc, aby funkcja mogła zmienić więcej obiektów, należy ją wywołać z argumentami przekazanymi przez wskaźnik lub referencję.

Przykład:

```
int zmien_wartosc(int aa, int *bb, int &cc)  //definicja funkcji
{
   aa=aa+100;
   *bb=*bb+100;
   cc=cc+100;

   return aa;
}

main()                                  //definicja funkcji main
{
   int a=5, b=10, c=15, d;              //definicje zmiennych

   cout<<a<<b<<c<<d;

   d=zmien_wartosc(a,&b,c);             //wywołanie funkcji

   cout<<a<<b<<c<<d;
}
```

przez wartość

przez wskaźnik

przez referencję

Przed wywołaniem funkcji: a=5, b=10, c=15, d=? (śmieć)
Po wywołaniu funkcji: a=5, b=110, c=115, d=105

Przekazywanie tablic do funkcji

Przekazywanie tablic do funkcji

C++ automatycznie (domyślnie) przekazuje całe tablice do funkcji używając **przekazywania przez referencję**, tj. wywołana funkcja może modyfikować wartości elementów w oryginalnych tablicach.
Pojedyncze elementy tablicy automatycznie są **przekazywane przez wartość**.
Tablicę w funkcji można odebrać na 2 sposoby: **1) jako tablicę, b) jako wskaźnik**.

Przykład:

```
int tablica[24];
void modyfikuj_tablice_ref(int tab[],int rozmiar);      //odbieramy tablicę jako tablicę
void modyfikuj_tablice_wsk(int *tab,int rozmiar);       //odbieramy tablicę jako wskaźnik
void modyfikuj_element(int element);

main()
{
   modyfikuj_tablice_ref(tablica,24);          //przekazywanie przez referencję
   modyfikuj_tablice_wsk(tablica,24);          //przekazywanie przez wskaźnik
   modyfikuj_element(tablica[5]);              //przekazywanie przez wartość
}

void modyfikuj_tablice_ref(int tab[],int rozmiar)
{
   tab[5]=tab[5]+2;          //modyfikacja oryginalnej tablicy!!!
}
void modyfikuj_tablice_wsk(int *tab,int rozmiar)
{
   tab[5]=tab[5]+2;          //modyfikacja oryginalnej tablicy!!!
   *(tab+5)+=2;              //modyfikacja oryginalnej tablicy!!!
}
void modyfikuj_element(int element)
{
   element=element+2;    //modyfikacja kopii przekazanego elementu!!!
}
```

Kwalifikator `const` w połączeniu ze wskaźnikami

Istnieje sześć możliwości zastosowania modyfikatora **const** w połączeniu z parametrami funkcji – dwa podczas przekazywania parametrów przez wartość oraz cztery podczas przekazywania ich przez wskaźnik.

W jaki sposób wybrać tę właściwą? Powinna Ci pomóc zasada najmniejszych przywilejów:
zawsze należy pozwalać funkcji tylko na taki dostęp do swych parametrów, aby mogła wykonać zadanie, ale nie na większy.

Cztery sposoby przekazania wskaźnika do funkcji:

1. **zmienny wskaźnik do zmiennych danych (ZWZD):**
   to taki wskaźnik, za pomocą którego mogą być modyfikowane dane, modyfikowany może być też sam wskaźnik

2. **zmienny wskaźnik do stałych danych (ZWSD):**
   to taki wskaźnik, który może być tak zmodyfikowany, aby wskazywał dowolne dane odpowiedniego typu, jednak same dane nie mogą być za jego pomocą zmodyfikowane

3. **stały wskaźnik do zmiennych danych (SWZD):**
   to taki wskaźnik, który zawsze pokazuje na to samo miejsce w pamięci, dane znajdujące się pod tym adresem mogą być za jego pomocą modyfikowane (**tablica – nazwa tablicy jest stałym wskaźnikiem jej zerowego elementu**)

4. **stały wskaźnik do stałych danych (SWSD):**
   to taki wskaźnik, który zawsze wskazuje na to samo miejsce w pamięci, a znajdujące się tam dane nie mogą być modyfikowane

Przykład:

```
int a=4;                          //zmienna
const b=5;                        //stała

int *zwzd=&a;                     //zmienny wskaźnik do zmiennych danych
const int *zwsd=&b;               //zmienny wskaźnik do stałych danych
int * const swzd=&a;              //stały wskaźnik do zmiennych danych
const int * const swsd=&b;        //stały wskaźnik do stałych danych
```

Typowy błąd programisty

Nie zainicjalizowanie wskaźnika powoduje, że wskazuje on na nieznany lub nie zainicjalizowany obszar pamięci i może być przyczyną przypadkowych modyfikacji istotnych danych lub **krytycznych błędów logicznych**. Nie zainicjalizowanie wskaźnika zadeklarowanego jako `const` jest traktowane jako **błąd składniowy**.

Typowy błąd programisty

Próba dereferowania zmiennej nie będącej zmienną wskaźnikową jest **błędem składni**.

Typowy błąd programisty

Dereferowanie wskaźnika mającego wartość 0 (zero, NULL) jest **krytycznym błędem wykonania programu**.

Typowy błąd programisty

Wykorzystanie arytmetyki wskaźników (odejmowanie, dodawanie, porównywanie) do wskaźników nie mających oparcia w tej samej tablicy jest zwykle **błędem logicznym**.

Typowy błąd programisty

Przekroczenie górnej lub dolnej granicy tablicy jest zwykle **błędem logicznym**.

Typowy błąd programisty

Mimo że nazwy tablic są wskaźnikami do ich początku, a wskaźniki można modyfikować korzystając z wyrażeń arytmetycznych, nazwy tablic nie mogą być zmieniane w ten sposób, ponieważ są wskaźnikami stałymi.

Dobry styl programisty

Korzystaj z przekazywania parametrów przez wartość, o ile funkcja wywoływana wyraźnie nie wymaga modyfikowania wartości argumentów w środowisku funkcji wywołującej. Jest to kolejny przykład zasady ograniczonych przywilejów.

Wskazówka dotycząca przenośności

Format wykorzystywany do wyświetlenia wskaźnika jest zależny od typu komputera. Część systemów wyświetla je jako liczby szesnastkowe, inne jako liczby dziesiętne.

Wskazówka praktyczna

Gdy wykorzystywane jest przekazywanie parametrów przez wartość, w funkcji wywołującej możliwa jest modyfikacja tylko jednej zmiennej. Zmiennej tej musi być przypisana wartość zwracana przez funkcję. Jeżeli natomiast zmodyfikowanych ma być kilka wartości, powinny one być przekazane do funkcji przez wskaźnik lub referencję.

Wskazówka praktyczna

Kwalifikator `const` może być wykorzystany do wymuszenia zasady najmniejszego przywileju.

Wskazówka dotycząca wydajności

Jeżeli będziesz przekazywał duże obiekty do funkcji za pomocą wskaźników lub referencji do stałych danych, program osiągnie wydajność charakterystyczną dla przekazywania przez referencję oraz bezpieczeństwo typowe dla przekazywania przez wartość.

Podsumowanie

PODSUMOWANIE 1:

- Wskaźniki to zmienne, których wartość jest adresem innych zmiennych.
- Istnieją trzy wartości, które mogą być wykorzystane do inicjalizacji wskaźnika: 0, NULL oraz adres obiektu tego samego typu. Inicjalizacja wskaźnika wartością 0 lub NULL ma to samo znaczenie.
- Jedyną liczbą całkowitą, którą można przypisać wskaźnikowi jest 0.
- Operator & (adres) zwraca adres swego operandu.
- Operand operatora adresu & musi być nazwą zmiennej (obiektu); operator adresu nie może być zastosowany do nazwy stałej, do wyrażenia, które nie zwraca referencji oraz do zmiennej, która została zadeklarowan jako **`register`**.
- Próba modyfikacji wartości zmiennej zadeklarowanej jako **`const`** zostaje wychwycona przez kompilator i zostanie zgłoszony komunikat o ostrzeżeniu lub błędzie.
- Tablice są automatycznie przekazywane jako argumenty funkcji przez referencje z wykorzystaniem wskaźników, ponieważ wartość nazwy tablicy jest jej adresem (adresem do jej pierwszego elementu).
- Aby przekazać jeden element tablicy przez referencję używając wskaźników, musi być przekazany adres danego elementu tablicy.
- Operacje arytmetyczne, które mogą być wykonywane na wskaźnikach to zwiększenie (++), zmniejszenie (--), dodanie/odjęcie do/od wskaźnika liczby całkowitej (+ lub +=, - lub -=) oraz odejmowanie/dodawanie jednego wskaźnika od/do drugiego.
- Operacje arytmetyczne na wskaźnikach powinny być wykonywane na ciągłych blokach pamięci, takich jak tablice.
- Wskaźniki mogą być sobie wzajemnie przypisywane, jeżeli są tego samego typu.

PODSUMOWANIE 2:

- Wskaźniki mogą być porównywane z wykorzystaniem równości oraz innych operatorów relacyjnych. Porównywanie to ma sens tylko wtedy, gdy oba wskaźniki wskazują na elementy tej samej tablicy.

- W połączeniu ze wskaźnikami mogą być stosowane indeksy, w taki sam sposób jak w przypadku tablic.

- W notacji wskaźnik/przesunięcie, przesunięcie jest odpowiednikiem indeksu w tablicy.

- Nazwa tablicy jest stałym wskaźnikiem, który zawsze wskazuje na to samo miejsce w pamięci (na jej pierwszy element).

- Możliwe jest tworzenie tablic wskaźników oraz wskaźników do funkcji. Wskaźnik do funkcji to adres, pod którym widnieje kod tej funkcji.

Ćwiczenia powtórzeniowe

1. Wykonaj poniższe polecenia:

a) podaj przykład dwóch różnych wyrażeń, przypisujących zmiennej wskaźnikowej adres tablicy zmiennoprzecinkowej

ODP: `float *wsk=tablica;` **ODP**: `float *wsk=&tablica[0];`

b) wydrukuj zawartość tablicy korzystając z a) notacji wskaźnik/przesunięcie i wskaźnika b) notacji wskaźnik/przesunięcie, jeżeli funkcję wskaźnika pełni nazwa tablicy oraz c) indeksując wskaźnik

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

ODPa:
```
{  cout<<*(wsk+i)<<' ';
```
ODPb:
```
   cout<<*(tablica+i)<<' ';
```
ODPc:
```
   cout<<wsk[i]<<' ';   }
```

c) napisz nagłówek (prototyp) funkcji pobierającej jako parametry dwa wskaźniki do liczb zmiennoprzecinkowych, która nie zwraca żadnej wartości

```
void f(float *x, float *y);
```

2. Określ, które z poniższych zdań są prawdziwe, a które fałszywe:

a) operator adresu & może być stosowany jedynie do stałych, wyrażeń oraz do zmiennych zadeklarowanych jako `register`

FAŁSZ: Operator adresu może być stosowany jedynie do zmiennych. Nie może natomiast być wykorzystany do stałych, wyrażeń oraz zmiennych `register`.

b) wskaźnik zadeklarowany jako `void` może być dereferowany

FAŁSZ: Wskaźnik `void` nie może być dereferowany, ponieważ nie istnieje możliwość dokładnego określenia, ile bajtów pamięci powinno być dereferowane.

c) wskaźniki różnych typów mogą być sobie wzajemnie przypisywane bez konieczności konwersji typów

FAŁSZ: Wskaźniki innych typów mogą być przypisywane jedynie wskaźnikom `void`. Wskaźniki `void` mogą być przypisywane innym wskaźników po dokonaniu jawnej konwersji typów.

Ćwiczenia sprawdzające

1. Określ, które z poniższych zdań są prawdziwe:
   a) dwa wskaźniki wskazujące na różne tablice nie mogą być ze sobą porównywane
   b) ponieważ nazwa tablicy jest wskaźnikiem do jej pierwszego elementu, można na niej wykonywać dokładnie takie same operacje, jak na wskaźniku

2. Napisz program do tasowania i wybierania kart. Program powinien wydawać dwa zestawy po pięć kart (gra w pokera), wyznaczać ich „wartość” i określać, który zestaw jest „mocniejszy”. Program powinien mieć możliwość określania, czy w zestawie jest para, dwie pary, trójka, kolor itp. (UWAGA: to zadanie jest dość trudne).

Następny wykład

Wykład nr 4

Temat: Dynamiczny przydział pamięci, zastosowania wskaźników, praca z plikami.