Programowanie 1

Sławomir Pluciński (splucinski@pjwstk.edu.pl)

Wskaźniki

- Przechowuje **adres** obiektu wskazanego typu.
- Wskaźnik może wskazywać na obiekt dowolnego typu.

Przykład:

```
int *w;
char *w_char;
float *w_float;
```

Do danego obiektu można odnosić się na dwa sposoby:

- Do nadawania wskaźnikowi wartości początkowej służy jednoargumentowy operator &.
- Oblicza on adres obiektu, który stoi po prawej stronie operatora przypisania.

Przykład:

```
int *w;      // definicja wskaźnika do obiektu typu int
int k = 10;     // definicja i inicjalizacja zmiennej int

w = &k;      // ustawienie wskaźnika na obiekt k
cout << *w; // wypisania wartosci obiektu *w
```

Do danego obiektu można odnosić się na dwa sposoby:

- Przez jego nazwę
- Przez zdefiniowany wskaźnik, który wskazuje na ten obiekt.

Operacja odniesienia się * do danego obiektu może być zarówno po prawej jak i po lewej stronie operacji przypisania.

Przykład:

```
w = &k;

k = *w + 10
*w = 10 + k;
```

Wskaźnik a referencja

```
int main() {
    int a = 1;
    int *adr;
    int &r_a = a;
    adr = &a;
    cout << * adr << endl;
    cout << &r_a << endl;
    cout << &a << endl;
    cout << adr << endl;
    cout << a << endl;
    cout << r_a << endl;
    r_a = 2;
    adr = &r_a;
    cout << adr << endl;
    cout << *adr << endl;
}
```

```
1
0x6ffe0c
0x6ffe0c
0x6ffe0c
1
1
0x6ffe0c
2
```

Wskaźnik typu **void**:

Wskaźnik jest to adres miejsca w pamięci plus informacja o tym, na jakiego typu obiekt wskazuje.

Można jednak zdefiniować wskaźnik bez wskazania typu:

```
void *wv;
```

gdzie *wv jest wskaźnikiem dowolnego typu void.

Przykład:

```
int *wi1, *wi2;
float *wf;

wi1 = wi2;
wf = wi1     //błąd

wf = (float *)wi1;
```

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

wv = wf;
wv = wch;
wv = wi;
```

- Wskaźnik każdego niestałego typu można przypisać do wskaźnika typu **void**
- Wskaźnika typu **void** nie można przypisać wskaźnikowi „prawdziwemu”
- Trzeba w takich sytuacjach posłużyć się operatorem rzutowania:

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

Wskaźniki do tablic

```
int *w       // definicja wskaźnika na obiekty typu int
int tab[20] //definicja tablicy typu int
w = &tab[i] // ustawienie wskaźnika na i-ty element tablicy

w = &tab[0]
w = tab

w = w + 1 // inaczej w++;
w += n    // inaczej w = w + n;
```

Wskaźniki do tablic

Zapisy równoważne:

```
w = &tab[0];
w = tab;

tab + 4;
&tab[4]

tab[5]
*(Tab+5)
```

- Wskaźniki do tablic można od siebie odejmować
- Odjęcie od siebie dwóch wskaźników pokazujących na różne elementy tej samej tablicy daje w wyniku liczbę dzielących je elementów tablicy.
- Liczba ta może być dodatnia lub ujemna.

Wskaźniki można też ze sobą porównywać za pomocą operatorów relacji

Wskaźniki do obiektu const

Jeśli funkcja otrzymuje adres tablicy, to pracuje w tym przypadku na oryginale tablicy i może dowolnie zmieniać wartości jej elementów.

- Jeżeli nie chcemy, aby funkcja zmieniła wartości to należy posłużyć się wskaźnikiem do stałego obiektu.
- Taki wskaźnik wskazuje na obiekt ale nie pozwala na jego modyfikacje.

Rezerwowanie obszarów pamięci

- Rezerwowanie i zwalnianie obszarów pamięci jest wykonywane za pomocą operatorów **new** i **delete.**
- Operator **new** zajmuje się tworzeniem, a delete destrukcją obiektów.

Przykład:

```
char *wchar;
wchar = new char; // utworzenie nowego obiektu typu char
delete wchar;     // zlikwidowanie tego obiektu

float *w;
w = new float[10];
delete [] w;
```

Cechy obiektów stworzonych operatorem new

- Obiekty takie istnieją od momentu, gdy je stworzyliśmy operatorem **new** do momentu, gdy je skasujemy operatorem **delete**. Więc, programista decyduje o czasie ich życia.
- Obiekt tak utworzony nie ma nazwy. Można nim operować tylko za pomocą wskaźników.
- Obiektów tych nie obowiązują zwykłe zasady o zakresie ważności. Jeśli tylko w danej chwili jest dostępny choćby jeden wskaźnik, który na taki obiekt wskazuje, to mamy dostęp do tego obiektu.
- Tylko statyczne obiekty są inicjalizowane wstępnie zerami. Natomiast obiekty utworzone operatorem new po utworzeniu przechowują wartości przypadkowe.

Dynamiczna alokacja pamięci na potrzeby tablicy

```
int *w_tab;
w_tab = new int[rozmiar];
```

- rozmiar jest wyrażeniem typu int.
- Została stworzona nienazwana tablica elementów typu int.
- Wynikiem działania operatora **new** jest wskaźnik do początku tej tablicy
- Tablica definiowana jest dynamicznie w trakcie wykonywania programu

```
cout << "Ile liczb w tablicy: ";
cin >> rozmiar;

int *w_tab;
w_tab = new int[rozmiar];

*w_tab = 50;          // wpisanie do zerowego elementu liczby 50
w_tab[0] = 50;

*(w_tab + 4) = 100; //wpisanie do elementu o indeksie 4 wartości 100
w_tab[4] = 100;

delete [] w_tab;
```

Stałe wskaźniki

- Dotychczas mówiliśmy o wskaźnikach do obiektów stałych.
- Są to wskaźniki, które nie mogą zmieniać pokazywanego obiektu. Traktują go jako obiekt stały.
- Sam obiekt, na który pokazują nie musi być rzeczywiście stały.
- Ważne jest to, że wskaźnik tak go traktuje.

```
int dworzec;
int *const wi = &dworzec
```

Stałe wskaźniki, a wskaźniki do stałych

Dotychczas mówiliśmy o wskaźnikach do obiektów stałych.

- Stały wskaźnik to taki, który zawsze pokazuje to samo. Nie można nim poruszyć.
- Wskaźnik do stałego obiektu to taki wskaźnik, który uznaje pokazywany obiekt za stały. Nie można modyfikować jego wartości.

Te dwa typy wskaźników można ze sobą łączyć:

```
const float *const wf = &a
```

Sposoby ustawienia wskaźników

- Wskaźnik można ustawić tak, aby pokazywał na jakiś obiekt, wstawiając do niego adres wybranego obiektu:

```
w = &obiekt;
```

- Wskaźnik można również ustawić na to samo, na co pokazuje już inny wskaźnik. Jest to zwykła operacja przypisania wskaźników:

```
w = w_new;
```

- Wskaźnik ustawia się na początek tablicy podstawiając do niego jej adres. W zapisie jest niepotrzebny operator &:

```
w = tab;
```

- Wskaźnik może także pokazywać na funkcję. Nazwa funkcji jest także jej adresem, zatem i tu zbędny jest operator &:

```
w = funkcja();
```

- Operator **new** zwraca adres nowoutworzonego obiektu. Taki adres wpisujemy do wskaźnika. Od tej pory wskaźnik pokazuje na ten nowy obiekt:

```
float *wf;
wf = new float;
```

Zastosowanie wskaźników

- Wskaźniki zwiększają efektywność pracy z tablicami.
- Dynamiczne rezerwowanie i zwalnianie obszarów pamięci.
- Przekazywanie w ramach funkcji oryginałów zmiennych
- Możliwość współpracy z urządzeniem zewnętrznym (np. miernikiem)