Wskaźniki

Istnienie wskaźników nie jest konieczne, dowodem na to są języki programowania, w których wskaźników nie ma. Z drugiej jednak strony - jeśli język C++ ma opinię tak sprawnego, to jest to głównie za sprawą wskaźników.

- Wyobraźmy sobie, że mamy w ręce bardzo gruby rozkład jazdy, np. 600 stron.
- Ktoś pyta nas o pociąg z Gdańska do Warszawy.
- Otwieramy więc rozkład jazdy i szukamy, przewracając strony. Wreszcie po dłuższym czasie trafiamy na właściwą i przebiegając palcem kolumny cyfr znajdujemy godzinę 8:05. Zamykamy książkę.
- Wtedy ten ktoś pyta: "A następny?".
- Bierzemy znowu rozkład jazdy, przewracamy strony, trafiamy na właściwą, potem znowu przebiegamy kolumny i odczytujemy - 9:10.
- Tylko że teraz na wszelki wypadek trzymamy palec na znalezionej godzinie odjazdu. Jeśli ten ktoś zapyta nas o następny pociąg, błyskawicznie przesuniemy palec na kolejną kolumnę i odczytamy godzinę. A jeżeli zapyta o godzinę przyjazdu pociągu do Warszawy, wtedy przesuniemy palec na wiersz poniżej i od razu udzielimy odpowiedzi.

Podany przykład ilustruje sytuację ze wskaźnikami i bez.
Wskaźnikiem w tym przypadku był nasz palec ale również wiedza o tym, jak należy go przesunąć w przypadku pytania: "A następny?".
Takich wskaźników możemy wykorzystywać dowolnie wiele, w przypadku książki z rozkładem jazdy można o nich myśleć jak o zakładkach.

Definiowanie wskaźników

Definicja

```
int *w;
```

mówi, że **w** jest wskaźnikiem do pokazywania na obiekty typu `int`.

- Gwiazdka * informuje, że mamy do czynienia ze wskaźnikiem.
- Nazwa wskaźnika to **w** (bez gwiazdki).
- We wskaźniku **w** możemy przechowywać adres jakiegoś obiektu typu `int`.
- Treścią wskaźnika jest informacja o tym, gdzie wskazywany obiekt się znajduje, a nie co się w nim znajduje.
- W definicji utworzyliśmy wskaźnik **w**, ale nie ustawiliśmy go jeszcze na żaden obiekt. Jednym z najczęściej popełnianych błędów ze wskaźnikami jest brak początkowego ustawienia wskaźnika. Przy braku ustawienia następuje odczyt z przypadkowego miejsca w pamięci.

Definiowanie wskaźników

Definicja

```
int *a, b, c;
```

oznacza, że definiujemy jeden wskaźnik (a) do obiektów typu `int` oraz dwie zwykłe zmienne (b, c) typu `int`.

Aby utworzyć trzy wskaźniki, należy napisać gwiazdkę przy każdej nazwie:

```
int *a, *b, *c;
```

Wskaźnik służący do pokazywania na obiekty jednego typu nie nadaje się do pokazywania na obiekty innego typu.
Np. wskaźnikiem typu `int` nie można pokazywać na obiekt typu `float`.
Próba ustawienia wskaźnika na obiekt innego, niewłaściwego typu spowoduje błąd kompilacji.

Praca ze wskaźnikiem

```
#include <iostream>
using namespace std;
int main()
{
   int k = 5;   //utworzenie zwykłej zmiennej typu int
   int *w;      //utworzenie wskaźnika typu int
  w = &k;
   cout << "Wskaźnik w pokazuje na obiekt o adresie " << w << endl;
   cout << "W obiekcie pokazywanym przez wskaźnik w "
        << "jest wartość " << *w;
}
```

```
Wskaźnik w pokazuje na obiekt o adresie 0x6ffe34
W obiekcie pokazywanym przez wskaźnik w jest wartość 5
```

- Operator **&** (*ampersand*) zwraca adres obiektu `k`. Adres ten następnie zostaje przypisany do wskaźnika `w`. Od tej pory wskaźnik `w` pokazuje na miejsce w pamięci, gdzie znajduje się obiekt `k`.
- Aby dostać się do wartości obiektu `k`, na który pokazuje wskaźnik `w`, musimy użyć operatora wyłuskania * (*gwiazdka*).
- Instrukcje `cout << k;` i `cout << *w;` są równoważne.

Praca ze wskaźnikiem

```
#include <iostream>
using namespace std;
int main()
{
   int a = 2, b = 5;

   int *w;
   w = &a;              //lub w skrócie: int *w = &a;

   cout << "a = " << a << "\t   *w = " << *w << endl;
   a = 10;
   cout << "a = " << a << "\t   *w = " << *w << endl;
   *w = 200;
   cout << "a = " << a << "\t   *w = " << *w << endl;
   w = &b;
   cout << "a = " << a << "\t   *w = " << *w << endl;
}
```

```
a = 2       *w = 2
a = 10      *w = 10
a = 200     *w = 200
a = 200     *w = 5
```

Praca ze wskaźnikiem

Jak widzieliśmy na poprzednim slajdzie: do obiektu można coś wpisać, używając jego nazwy, albo wskaźnika, który na ten obiekt pokazuje:

```
int  a = 2;
int *w = &a;

 a = 3;   //wpisujemy do zmiennej a wartość 3, używając jej nazwy
*w = 3;   //wpisujemy do zmiennej a wartość 3, używając wskaźnika, który na nią pokazuje
```

Poniżej inny przykład równoważnych instrukcji:

```
int  a = 2;
int *w = &a;
int  b;

b =  a;   //wpisujemy do b wartość zmiennej a
b = *w;   //wpisujemy do b wartość zmiennej a, używając wskaźnika, który pokazuje na a
```

Przypisywanie wskaźnikowi innego wskaźnika:

```
int a = 2;
int *w, *u;
w = &a;
u = w;   //"niech wskaźnik u pokazuje na to samo co wskaźnik w"
```

Wskaźniki i rzutowanie

```
int a = 2;
int *w, *u;
w = &a;
u = w;      //"niech wskaźnik u pokazuje na to samo co wskaźnik w"
```

Przypisanie do wskaźnika `u` adresu wskaźnika `w` jest możliwe dzięki temu, że oba wskaźniki są tego samego typu (int). Jednak poniższy kod zakończy się błędem kompilacji:

```
int a;   float b;
int    *wi = &a;
float  *wf = &b;
wf = wi;        //błąd
wi = wf;        //błąd
```

Wskaźnik `wf` jest typu `float` i nie może pokazywać na to samo, co wskaźnik `wi` typu `int` (i na odwrót). Aby takie przypisanie było możliwe, potrzebny jest operator rzutowania:

```
int a;   float b;   char c;
int    *wi = &a;
float  *wf = &b;
char   *wc = &c;
wf = (float *) wi;   //rzutowanie
wc = (char *) wi;    //rzutowanie
```

Wskaźnik typu void

Wiemy już, że

Wskaźnik to adres pewnego miejsca w pamięci oraz informacja o typie obiektu, na który może on pokazywać.

Możemy jednak zdefiniować wskaźnik bez informacji o typie.
Mówimy wtedy, że jest to wskaźnik typu **void**.
Do takiego wskaźnika można przypisać wskaźnik dowolnego (niestałego) typu bez konieczności rzutowania:

```
int a;   float b;   char c;
int    *wi = &a;
float  *wf = &b;
char   *wc = &c;
void   *wv;
wv = wi;
wv = wf;
wv = wc;
void *wv2;
wv2 = wv;
wi = (int *) wv;   //ale tutaj już konieczne rzutowanie
```

Wskaźnik typu void

Wskaźnik void, pozbawiony informacji o typie, nie może służyć do odczytania miejsca, na które pokazuje. Nie można też poruszać nim po sąsiednich obiektach (gdy mamy je zebrane w tablicę).

Dlaczego w takim razie wymyślono taki wskaźnik?

Czasami wiedza o typie pokazywanego obiektu była niepotrzebnym balastem. Odczuwało się to szczególnie w starszych funkcjach bibliotecznych, do których trzeba wysłać (lub odebrać) adres jakiegoś obiektu w pamięci. Funkcję taką nie interesuje co tam jest, a jedynie gdzie to jest. Odbiera ona (lub zwraca) ten adres za pomocą wskaźnika `void`.

Stara funkcja `malloc()` dawała adres zarezerwowanego miejsca w pamięci jako wskaźnik `void`, a my już musieliśmy zatroszczyć się o resztę.

Wskaźniki i tablice

```
int tab[10];
int *w;
w = &tab[5]; //wskaźnik w ustawiamy na elemencie o indeksie 5 tablicy
```

```
int tab[10];
int *w;
w = &tab[0];   //wskaźnik w ustawiamy na elemencie o indeksie 0
w = tab;       //to samo, bo nazwa tablicy jest adresem jej początku
```

Nazwa tablicy jest jednocześnie adresem jej początku.

```
int tab[10];
int *w = tab;   //wskaźnik w pokazuje na początek tablicy tab
w++;            //a teraz na element o indeksie 1
w = w + 3;      //a teraz przesunął się o 3 miejsca dalej,
                //czyli na element o indeksie 4
w = w - 1;      //teraz pokazuje na element o indeksie 3
```

Do wskaźników pokazujących na tablicę możemy zatem dodawać (odejmować) liczby naturalne. Spowoduje to przesunięcie wskaźnika o zadaną liczbę komórek tablicy do przodu (do tyłu).

Wskaźniki i tablice

Trzy sposoby wypisania elementu tablicy:

```
int tab[5] = {1,2,3,4,5};
cout << tab[3];      //bez wskaźnika
int *w = &tab[3];    //tworzymy nowy wskaźnik
cout << *w;
cout << *(tab+3);    //wykorzystujemy nazwę tablicy
```

Zapis **tab+3** oznacza to samo co **&tab[3]**.

Zapis ***(tab+3)** oznacza to samo co **tab[3]**.

Przykład błędnej operacji na nazwie tablicy:

```
int tab[5] = {1,2,3,4,5};
int *w = tab;
w++;    //OK
tab++;  //błąd, ten wskaźnik nie może się przesuwać
```

Nazwa tablicy jest stałym wskaźnikiem do jej początku.

Wskaźniki i tablice

Wskaźniki (pokazujące na tę samą tablicę) można odejmować:

```
#include <iostream>
using namespace std;
int main()
{
   int tab[15];
   int *w_a = &tab[3];
   int *w_b = &tab[9];
   int *w_c = &tab[10];
   cout << w_b - w_a << endl;
   cout << w_c - w_b << endl;
   cout << w_a - w_c << endl;
   cout << w_c - w_a << endl;
}
```

Wynik działania programu:

```
6
1
-7
7
```

Wskaźniki i tablice

Wskaźniki (pokazujące na tę samą tablicę) można porównywać:

```
#include <iostream>
using namespace std;
int main()
{
   int tab[5], i;
   int *w = &tab[3];
   int *v;
   cout << "Wskaźnik w pokazuje na element o indeksie 3." << endl;
   cout << "Na który element ma pokazywać wskaźnik v? (0-4): ";
   cin >> i;
   if (i>=0 && i<=4)
   {
      v = &tab[i];
      if (v < w)
         cout << "v pokazuje na element bliżej początku tablicy niż w";
      else if (v > w)
         cout << "v pokazuje na element o wyższym indeksie niż w";
      else cout << "v pokazuje na to samo co w";
   }
}
```

```
#include <iostream>
using namespace std;
int main()
{
   int tab[5];
   int *w = tab;
   for (int i = 0; i < 5; i++)
   {
      cout << "Podaj kolejny element tablicy: ";
      cin >> *w;
     w++;
   }
   w = tab;
   cout << "Elementy tablicy: ";
   for (int i = 0; i < 5; i++)
   {
      cout << *(w + i) << " ";
   }
   w = tab;
   int s = 0;
   for (int i = 0; i < 5; i++, w++)
   {
      s = s + *w;
   }
   cout << endl << "Suma elementów tablicy: " << s;
}
```

Stałe wskaźniki

Wiemy już, że przykładem stałego wskaźnika jest nazwa tablicy.
Stały wskaźnik to taki, który zawsze pokazuje na to samo, nie można go przesuwać.

Definicja stałego wskaźnika:

```
int n;
int * const w = &n;
```

Definicja niepoprawna:

```
int n;
int * const w;
w = &n;              //błąd
```

Przykładowe operacje:

```
int n, m;
int * const w = &n;
w++;                 //błąd
w = &m;              //błąd
*w = 5;              //OK
```

Wskaźniki do obiektów stałych

Są to wskaźniki, które pokazywanego obiektu nie mogą zmieniać. Traktują go jako obiekt stały.
Sam obiekt, na który pokazują, nie musi być rzeczywiście stały. Ważne jest to, że wskaźnik tak go traktuje.

Definicja wskaźnika do obiektu stałego:

```
int n;
const int *w;
w = &n;
```

Przykładowe operacje:

```
int n, m, k;
const int *w = &n;
*w = 5;          //błąd, bo w traktuje n jako obiekt stały
n = 5;           //OK, bo n nie jest obiektem stałym
*w = *w + 1      //błąd
m = *w;          //OK, wskaźnik w może odczytywać zawartość obiektu, na
w = &k;          //OK, wskaźnik w można przesuwać
int *v = &n;
(*v)++;          //OK, bo v jest zwykłym wskaźnikiem
```

Stałe wskaźniki do obiektów stałych

Są to stałe (nieruchome) wskaźniki do stałych (niezmiennych) obiektów.

Definicja stałego wskaźnika do stałych:

```
int n;
const int * const w = &n;
```

Przykładowe operacje:

```
int n, m, k;
const int * const w = &n;
m = *w;   //OK, odczytanie wartości z obiektu pokazywanego
*w = 5;   //błąd, wskaźnik w traktuje n jako obiekt stały
w = &n;   //błąd, wskaźnik w jest nieruchomy i zawsze pokazuje na n
```

Wskaźniki - podsumowanie

1. Zdefiniuj zmienną **n** typu **int** o wartości 3 oraz wskaźnik **wsk** do obiektów typu **int**. Ustaw wskaźnik na obiekt **n**.

2. Za pomocą wskaźnika **wsk** z zad. 1 wypisz zawartość i adres obiektu, na który wskaźnik **wsk** pokazuje.

3. Zdefiniuj inny wskaźnik do obiektów typu **int** i ustaw go na to samo, na co wskazuje wskaźnik **wsk** z zad.1 (na dwa sposoby).

4. Zdefiniuj wskaźnik do obiektów typu **float** i ustaw go na to samo, na co wskazuje wskaźnik **wsk** z zad.1.

5. Zdefiniuj wskaźnik typu **void** i ustaw go na to samo, na co wskazuje wskaźnik **wsk** z zad.1.

6. Zdefiniuj wskaźnik do obiektów typu **char** i ustaw go tak samo, jak zdefiniowany w zad.5 wskaźnik typu **void**.

7. Zdefiniuj tablicę **A** typu **int** o rozmiarze 10 oraz wskaźnik odpowiedniego typu i ustaw go na początku tej tablicy (na dwa sposoby).

8. Zdefiniuj tablicę **A** typu **int** o rozmiarze 10 oraz odpowiedni wskaźnik i ustaw go na piątym elemencie tej tablicy, tzn. na elemencie o indeksie 4 (na dwa sposoby).

9. Wskaźnik z zad.8 przesuń o 3 elementy dalej w tablicy (na dwa sposoby).

10. Do elementu zerowego tablicy **A** wpisz wartość 6 za pomocą wskaźnika (na dwa sposoby).

11. Do elementu o indeksie 5 tablicy **A** wpisz wartość 6 za pomocą Wskaźnika (na dwa sposoby).

12. Zdefiniuj tablicę typu **int** o rozmiarze 20 oraz dwa wskaźniki: pierwszy ustaw na szóstym elemencie tablicy a drugi na piętnastym. Jaki wynik otrzymamy odejmując te wskaźniki? Co on oznacza?

13. Zdefiniuj wskaźnik do obiektu stałego typu **float** oraz podaj przykład niedozwolonej operacji na tym wskaźniku.

14. Zdefiniuj stały wskaźnik do obiektu typu **float** oraz podaj przykład niedozwolonej operacji na tym wskaźniku.

15. Zdefiniuj stały wskaźnik do stałego obiektu typu **float** oraz podaj dwa przykłady niedozwolonych operacji na tym wskaźniku.