Rozdział 4

Definiowanie i praca ze wskaźnikami. Zastosowanie wskaźników

4.1 Wskaźniki

W języku C++ oprócz zwykłych zmiennych, które poznaliśmy do tej pory istnieją także tzw. wskaźniki. Aby nie mieć z nimi problemów warto zapamiętać następującą regułkę:

Zapamiętaj

Wskaźnik to zmienna służaca do przechowywania adresu obiektu.

Zaletą wskaźników jest to, że znacznie przyspieszają pracę z tablicami. Wskaźnik możemy ustawić na dowolną zmienną. Wpisujemy w ten sposób adres tej zmiennej/obiektu do tego wskaźnika. Wskaźniki definiujemy za pomocą operatora "gwiazdki" `*`, który stawiamy przed nazwą wskaźnika. W definicji wskaźnika oprócz jego nazwy należy na samym początku podać też typ zmiennej/obiektu na jaki może pokazywać tworzony wskaźnik. Listing 4.1 pokazuje przykład definiowania wskaźników.

```
1 int *wskInt;
2 double *a;
3 char *t;
```

Listing 4.1: Przykład definiowania wskaźnika

Definicje takie czytamy: `wskInt` jest wskaźnikiem do obiektów typu `int`, `a` jest wskaźnikiem do obiektów typ `double`, `t` jest wskaźnikiem do obiektów typu `char`, itp. Wskaźnik mogący pokazywać na `int` nie nadaje się do pokazywania na `double`.

Zapamiętaj

Wskaźnik mogący pokazywać na dany typ nie może pokazywać na inny typ.

W celu wpisania jakiegoś adresu do wskaźnika w programie muszą wcześniej istnieć jakieś obiekty. Pamiętamy, że do uzyskiwania adresu danej zmiennej służy operator `&` stawiany przed nazwą zmiennej. Sposób wpisania adresu zmiennej do wskaźnika przedstawia listing 4.2. Jak widać na jedną zmienną może pokazywać wiele wskaźników.

```
double liczba = 12.4; // utworzenie zmiennej typu double
double *w;            // utworzenie wskaznika do double
w = &liczba;          // wpisanie adresu zmiennej do wskaznika
double *w2 = &liczba; // utworzenie wskaznika i jednoczesne
                      // wpisanie adresu zmiennej
```

Listing 4.2: Przykład wpisania adresu zmiennej do wskaźnika

Za pomocą wskaźnika możemy odnosić się do zmiennej, na którą on pokazuje. Możemy nawet zmieniać wartość tej zmiennej. W celu odniesienia się do wartości jaka jest w zmiennej pokazywanej przez wskaźnik używamy znowu operatora gwiazdki `*` . Możemy to interpretować w ten sposób, że jeśli mamy zdefiniowany wskaźnik `wsk` to sam zapis `wsk` oznacza adres na jaki pokazuje wskaźnik. Natomiast zapis `*wsk` oznacza **obiekt**, na który pokazuje wskaźnik. Listing 4.3 pokazuje sposób w jaki można sprawdzić za pomocą wskaźnika wartość przechowywną w zmiennej.

```
int zz = 4;           // utworzenie zmiennej typu int
int *w;               // utworzenie wskaznika do int
w = &zz;              // wpisanie adresu zmiennej do wskaznika
cout << *w << endl;   // wyswietlenie wartosci zmiennej
                      // za pomoca wskaznika
*w = 77;              // wpisanie nowej wartosci do zmiennej
                      // za pomoca wskaznika
int aa = 33;          // definicja innej zmiennej int
w = &aa;              // przestawienie wskaznika na inna zmienna
```

Listing 4.3: Przykład odniesienia się do wartości zmiennej za pomocą wskaźnika

Przykład

Napisz program wypisujący wartość zmiennej na ekran w sposób "tradycyjny" i za pomocą wskaźnika. Następnie zmień wartość tej zmiennej za pomocą wskaźnika i sprawdź nową wartość zmiennej przez wypis na ekran. Sprawdź też jaki jest adres ukryty pod wskaźnikiem.

4.2 Adres zerowy - nullptr

Ważną rzeczą odnośnie wskaźników jest to, aby w momencie ich definicji je inicjalizować. Jeśli tego nie zrobimy to w przypadku, gdy wskaźnik jest obiektem lokalnym a nie globalnym (np. zdefiniowanym wewnątrz funkcji) to taki wskaźnik jest inicjalizowany przypadkowym adresem w pamięci komputera (wskaźnik globalny jest inicjalizowany adresem zero `nullptr`). Gdy wpiszemy do takiego lokalnego wskaźnika, którego nie inicjalizowaliśmy jakąś wartość to niszczymy tą informację, która była zapisana w komórce pamięci pokazywanej przez wskaźnik. Może to spowodować, że nasz program zacznie się bardzo dziwnie zachowywać, a znalezienie przyczyny będzie bardzo trudne. Listing 4.4 pokazuje przykład zniszczenia informacji przez nieustawienie wartości początkowej we wskaźniku.

```
1  void f()
2  {
3    double *x;        // brak inicjalizacji wskaznika
4                      // wskaznik zostaje ustawiony na przypadkowe
5                      // miejsce w pamieci komputera
6    *x = 45.5454;     // zniszczenie informacji w komorce pamieci
7                      // na ktora pokazuje wskaznik x
8  }
```

Listing 4.4: Przykład zniszczenia informacji w pamięci komputer przez niezainicjalizowanie wskaźnika

Aby temu zapobiec warto zapamiętać poniższą regułkę.

Zapamiętaj

Zawsze inicjalizuj wskaźnik. Jeśli nie ma on wartości początkowej to ustaw go na adres zerowy nullptr.

Listing 4.5 pokazuje jak wstawić adres zerowy do wskaźnika.

```
1  double *x = nullptr;
```

Listing 4.5: Przykład ustawienia do wskaznika adresu zerowego

Adres zerowy to specjalne miejsce, na które ustawiamy wskaźnik żeby zaznaczyć, że w danej chwili nie wskazuje on na żaden obiekt. Próba wpisania czegoś do takiego adresu nie powiedzie się i komputer rozpozna, że jest to adres zerowy. Przede wszystkim nie zostanie zniszczona informacja w przypadkowym miejscu w pamięci komputera. Zawsze można sprawdzić czy wskaźnik pokazuje na jakiś adres czy na adres zerowy. Przykład pokazuje listing 4.6.

```
1  double *x = nullptr;
2  if (x == nullptr) cout << "Adres zero." << endl;
3  else cout << "Inny adres." << endl;
```

Listing 4.6: Przykład sprawdzenia czy wskaźnik pokazuje na adres zerowy

Zadanie nr 1

Sprawdź co się stanie przy próbie wpisania wartości do wskaźnika pokazującego na adres zerowy.

4.3 Wskaźniki i tablice

Jeśli mamy zdefiniowany wskaźnik to możemy go ustawić na dowolny element tablicy co pokazano w listingu 4.7. Jak pamiętamy nazwa tablicy jest też adresem jej zerowego elementu. A więc wpisując do wskaźnika nazwę tablicy ustawiamy go na jej zerowy element.

```
double tab[20];
double *wsk;
wsk = tab;      // ustawienie wskaznika na element 0
wsk = &tab[4]; // ustawienie wskaznika na element 5
```

Listing 4.7: Ustawienie wskaźnika na element tablicy

Wskaźniki używamy często do pracy z tablicami, bo jest to najszybszy sposób. W tym celu ustawiamy wskaźnik na element zerowy i po wykonaniu wymaganych w naszym programie działań na tym elemencie przesuwamy wskaźnik o jeden element dalej itd. Przesunięcie wskaźnika, niezależnie od wielkości elementu, to po prostu dodanie do niego wartości 1. Listing 4.8 pokazuje jak poruszać wskaźnik po elementach tablicy. Przy poruszaniu wskaźnikami należy pamiętać żeby ustawiać je na istniejące elementy, bo inaczej dostaniemy bezsensowne wyniki lub, jeśli coś tam będziemy chcieli wpisać, zniszczymy jakąś informację zapisaną w pamięci komputera. Przykładowo jeśli mamy tablicę 100-elementową i ustawimy wskaźnik na element 101, który nie istnieje.

```
double tab[20];
double *wsk = tab;
wsk = wsk + 1; // przesuniecie wskaznika o 1 element
wsk++;         // przesuniecie wskaznika o 1 element
wsk += 10;     // przesuniecie wskaznika o 10 elementow
```

Listing 4.8: Przesuwanie wskaźnika po elementach tablicy

Przykład

Napisz program, w którym tworzona jest 10-elementowa tablica dla liczb rzeczywistych. Za pomocą wskaźnika wyświetl na ekranie adresy i wartości wszystkich elementów tablicy. Wykorzystaj pętlę oraz przesunięcie wskaźnika o jeden element ++.

4.4 Przesyłanie wskaźników do funkcji

Pamiętamy, że poznaliśmy dwa sposoby przesyłania argumentów do funkcji: przez wartość i przez referencję. W pierwszym przypadku do funkcji przesyłamy kopię danej zmiennej, a w drugim jej oryginał. Trzecim sposobem przesyłania argumentu do funkcji jest przesłanie przez wskaźnik. W tym sposobie do funkcji przesyłany jest, więc adres oryginału zmiennej za pomocą wskaźnika. Przykład deklaracji funkcji, która jako argument odbiera wskaźnik pokazuje listing 4.9.

```
1 double funkcja(double *wsk);
```

Listing 4.9: Przykład funkcji pobierającej argument jako wskaźnik

Przykład

Napisz funkcję, która pobiera jako argument adres do zmiennej typu całkowitego i do tej zmiennej dodaje wartość 1000. Wywołaj funkcję w programie i sprawdź przez wypisanie na ekran wartość zmiennej przed i po wywołaniu funkcji.

Jeśli pamiętamy, że nazwa tablicy jest jednocześnie adresem jej zerowego elementu to możemy napisać funkcję przyjmującą jako argument wskaźnik i przy wywołaniu funkcji podać jako argument nazwę tablicy.

Przykład

Napisz dwie funkcje, z których jedna przyjmuje jako argument tablicę jednowymiarową dla zmiennych double, a druga wskaźnik do double. Obie funkcje powinny robić to samo, czyli zmieniać wszystkie elementy tablicy na wartość 3.14. Utwórz w programie dwie takie same tablice z nadanymi wartościami początkowymi, wywołaj najpierw pierwszą funkcję na jednej tablicy, a potem drugą na drugiej tablicy. Wyświetl obie tablice przed i po wywołaniu funkcji na ekran. Uwaga: aby można zmienić wszystkie elementy tablicy funkcja musi wiedzieć jaki jest rozmiar tablicy, więc powinna też przyjmować tę wartość jako argument.

4.5 Zadania do rozwiązania

Zadania na ocenę 3,0

Zad. 1. Napisz program, który prosi użytkownika o wprowadzenie liczb całkowitych jako danych wejściowych, które będą przechowywane odpowiednio w zmiennych 'a' i 'b'. Istnieją również dwa wskaźniki całkowite o nazwach A i B. Przypisz wartości 'a' i 'b' odpowiednio do A i B i wyświetl je.

Zad. 2. Napisz program, który posiada tablicę T o 2000 elementów na liczby całkowite. Za pomocą wskaźnika wypełnij wszystkie elementy tablicy cyfrą 1.

Zadania na ocenę 3,5-4,0

Zad. 3. Napisz program, który oblicza sumę elementów danej tablicy. Zadanie wykonaj przy pomocy wskaźników.

Zadania na ocenę 4,5-5,0

Zad. 4. Napisz program podający maksymalną wartość znajdującą się w danej tablicy. Program zrealizuj przy użyciu wskaźników.

Zad. 5. Napisz program obliczający odchylenie standardowe dla danego zestawu danych. Program zrealizuj przy użyciu tablic i wskaźników.

Zadania nieobowiązkowe

Zad. 6. Napisz program, w którym uruchamiana jest funkcja przyjmująca jako argument wskaźnik do `char[]`. Funkcja powinna zwracać liczbę liter w przesłanym C-stringu. Wykorzystaj fakt, że C-string kończy się znakiem zerowym `null`. W programie C-string powinien być wprowadzany za pomocą klawiatury, a następnie powinna być wyświetlona informacja o długości C-stringu.

Bibliografia

[1] cplusplus.com. Variables and types. https://www.cplusplus.com/doc/tutorial/variables/.

[2] cpp0x.pl/ Serwis programistyczny C++. Biblioteka <math.h>. https://cpp0x.pl/kursy/Kurs-C++/Dodatkowe-materialy/Biblioteka-math-h/322.

[3] cpp0x.pl/ Serwis programistyczny C++. Pseudolosowe liczby całkowite. https://cpp0x.pl/kursy/Kurs-C++/Poziom-2/Pseudolosowe-liczby-calkowite/290.

[4] J. Grębosz. *Opus magnum C++11. Programowanie w języku C++*. Helion, 2019.