Rozdział 3

Praca ze zmiennymi tablicowymi statycznymi

3.1 Zmienne tablicowe

Mając pojedyncze dane do zapisania tworzymy sobie pojedyncze zmienne, np. `double a` i do takich zmiennych wpisujemy potrzebne dane. Często jednak zdarza się tak, że mamy bardzo wiele takich zmiennych tego samego typu. Przykładem może być ciąg liczb naturalnych od 0 do 100. Jeśli chcielibyśmy przechowywać te liczby w pojedynczych zmiennych to musielibyśmy 100 razy w programie to wpisać, a to nie byłoby takie proste. Poza tym zmiennych mogłoby być np. 10^6. W takich przypadkach bardzo przydają się zmienne tablicowe, czyli ciąg obiektów tego samego typu, które zajmują w sposób ciągły (obok siebie) miejsce w pamięci komputera. Obrazowo możemy sobie to wyobrażać jako szuflada z bardzo wieloma przegródkami, gdzie w każdej przegrodzie znajduje się inna wartość (inna zmienna).

Zaletą tablic jest to, że zamiast 100 czy 10^6 nazw zmiennych mamy jedną nazwę i do każdego elementu odnosimy się poprzez nawias kwadratowy i indeks, czli miejsce położenia danej zmiennej w tablicy. Listing 3.1 pokazuje przykład utworzenia zmiennej tablicowej do przechowywania 100 różnych zmiennych typu `double`. W linii 5 widać odniesienie do elementu tablicy nr 101. Taki element nie istnieje, bo tablica ma tylko 100 elementów. Jest to więc błąd, ale nie zostanie on zasygnalizowany przez kompilator. To co było w tym miejscu pamięci zostaje zniszczone. Jest to bardzo niebezpieczny błąd, bo jego rezultaty są często bardzo nieprzewidywalne i pojawiają się dopiero w momencie działania programu. Jeśli, więc program po uruchomieniu zachowuje się "dziwnie", a używaliśmy tablic to warto sprawdzić czy nie pomyliliśmy się w indeksach.

```
1 int tab[100];
2 tab[0] = 13;   // pierwszy element tablicy
3 tab[29] = 11;  // trzydziesty element tablicy
4 tab[99] = 16;  // ostatni (100) element tablicy
5 tab[100] = 19; // Blad! nie ma takiego elementu
```

Listing 3.1: Przykład utworzenia i użycia zmiennej tablicowej

Zapamiętaj

Zmienne tablicowe są numerowane w C++ od 0. Oznacza to, że w N-elementowej tablicy elementy są numerowane od 0 do N-1.

Ponadto należy pamiętać, że rozmiar tablicy (w Listingu 3.1 jest to liczba 100) musi być znany w chwili kompilacji. Nie można więc utworzyć takiej tablicy, gdzie chcielibyśmy np. żeby rozmiar tablicy został wczytany w czasie działania programu np. za pomocą komendy `cin >>` (Listing 3.2).

```
int rozmiar;
cin >> rozm
double t[rozmiar];  // Blad!
```

Listing 3.2: Błędny sposób tworzenia tablicy statycznej

Przykład

Napisz program tworzący tablice zmiennych typu `int` o rozmiarze 1000. Do pierwszego elementu wpisz 0, do drugiego 1, do trzeciego 2, ... Wypisz wartości elementów tablicy na ekran.

3.2 Inicjalizacja tablic

Wartości elementów tablicy można nadawać w momencie tworzenia tablicy, czyli możemy od razu zainicjalizować tablicę. Dla zwykłych zmiennych możemy (oprócz standardowego sposobu z operatorem =) możemy wykorzystać też składnię

```
int a{5};
double b{23.87};
int c{};            // inicjalizacja wartoscia 0
```

Listing 3.3: Inny sposób inicjalizacji zwykłych zmiennych

W przypadku tablic istnieje analogiczny sposób

```
int tabInt[1000]{};         // inicjalizacja elementow wartosicia 0
double tabDouble[1000]{};   // inicjalizacja elementow wartosicia 0.0
```

Listing 3.4: Inicjalizacja tablic zerami odpowiedniego typu

Aby zainicjalizować tablicę wartościami różnymi od 0 możemy skorzystać z trzech sposobów pokazanych w listingu 3.5.

```
int tab1[3] {1, 20, 30};
int tab2[40] = {10, 222, 32, 2};
int tab3[] = {10, 31, 32, 2000, 50, 76};
```

Listing 3.5: Inicjalizacja tablic wartościami różnymi od zera

W pierwszym sposobie widać utworzenie tablicy elementów `int` o rozmiarze 3 i następujących wartościach: `t[0] = 1, t[1] = 20, t[2] = 30`. W sposobie nr dwa tworzona jest tablica o rozmiarze 40, ale tylko pierwsze 4 elementy są inicjalizowane wartościami różnymi od zera, pozostałem wartością 0. W trzecim sposobie nie podaliśmy rozmiaru tablicy, a tylko wartości poszczególnych elementów. Dzięki temu kompilator sam wiedział, że rozmiar tablicy jest równy liczbie elementów więc utworzona tablica ma rozmiar 6.

Czasami zachodzi potrzeba utworzyć obiekt o stałej wartości, czyli taki którego nie można zmienić. W przypadku zwykłej zmiennej robi się to przez dodanie słowa kluczowego `const` przed zmienną. Należy także pamiętać, że w przypadku obiektu stałego obiekt ten musi zostać zainicjalizowany w momencie jego tworzenia. Podobne zasady obowiązują w przypadku tablic. Przykład inicjalizowania obiektów stałych pokazano w listingu 3.6.

```
const int a = 2;
const int b {22};
const int c[2] {22,33};
const double d[7] = {1,2,3,4,5,6,7};
```

Listing 3.6: Inicjalizacja stałego obiektu

3.3 Przekazywanie tablic do funkcji

Na poprzednich zajęciach widzieliśmy jak przekazywać zmienne do funkcji. Zmienna tablicowa też może zostać przesłana do funkcji lub inaczej funkcja może przyjmować jako argument tablicę. Należy jedynie pamiętać, że tablicy nie da się przesłać przez wartość. Tablicę przesyła się podając funkcji adres początku tej tablicy. Aby utworzyć funkcję, która jako argument ma pobierać tablicę należy to zaznaczyć przez dodanie nawiasów klamrowych jak w poniższym listingu 3.7.

```
double fun(double tab[]);
```

Listing 3.7: Przykład deklaracji funkcji pobierającej tablicę jako argument

Następnie wywołanie tej funkcji w kodzie wyglądałoby tak jak w listingu 3.8.

```
double tablica[4] = {10,20,30,40};
fun(tablica);
```

Listing 3.8: Przykład wywołania funkcji pobierającej tablicę jako argument

Jak widać wywołanie funkcji pobierającej tablicę nie zawiera nawiasów kwadratowych tej tablicy, a jedynie jej nazwę.

Zapamiętaj

Nazwa tablicy jest jednocześnie adresem jej zerowego elementu.

Czasem potrzeba też uzyskać adres zmiennej tablicowej. Możemy to zrobić tak jak dla zwykłej zmiennej tylko należy pamiętać o indeksie. Np. jeśli mamy

tablicę `double tab[20]` i chcemy się dowiedzieć jaki jest adres elementu nr 10 to powinniśmy napisać `&tab[9]`. Innym sposobem na to jest zapis `tab + 9`.

Przykład

Napisz program uruchamiający funkcję, która pobiera tablicę i: 1) wpisuje do kolejnych elementów tablicy liczby całkowite zaczynając od 0, 2) wyświetla wszystkie elementy z wpisanymi wartościami, 3) wyświetla adresy wszystkich elementów.

3.4 Tablice char

W programie możemy też tworzyć tablice do przechowywania znaków, czyli zmiennych typu `char`. Np. definicja `char nazwisko[30]` oznacza, że możemy w tej tablicy umieścić nazwiska o długości do 30 liter. Należy jednak pamiętać, że tekst w tablicach przechowuje się w ten sposób, że po ciągu liter następuje znak o kodzie 0, czyli `null`. Robi się to po to, aby program wiedział gdzie kończy się dane słowo. Taki ciąg znaków nazywamy *C-string*.

W celu inicjalizacji tablicy `char` wpisujemy tekst ujęty w cudzysłów jak pokazano na listingu 3.9.

```
1 char nazwisko[30] = {"Blasiak"};
```

Listing 3.9: Przykład inicjalizacji tablicy znakowej

Oznacza to, że w kolejnych elementach znajdą się znaki: B, l, a, s, i, a, k oraz 0. Pozostałe elementy też zostaną wypełnione 0 aż do końca tablicy.

Przykład

Napisz program sprawdzający rozmiar dwóch zmiennych: `char imie[]={'A','l','a'}` oraz `char imie2[]={"Ala"}`. Wykorzystaj do tego operator `sizeof`, (np. w postaci `cout << sizeof(imie)`). Zwraca on rozmiar zmiennej w bajtach. Jaki z tego wniosek?

3.5 Tablice wielowymiarowe

Tablica wielowymiarowa to tablica, której elementami są tablice. Zwykłe tablice możemy uważać jako jednowymiarowe. Tablice wielowymiarowe mają więcej niż jeden wymiar. Przykładem najprostszej tablicy wielowymiarowej jest tablica dwuwymiarowa, którą możemy sobie wyobrażać jako macierz lub tabelę (np. taką jak w MS Excel). Przykład definicji tablicy dwuwymiarowej o 1000 elementach, z których każdy element jest tablicą 100-elementową pokazuje listing 3.10. W tablicy dwuwymiarowej występuje dwie pary nawiasów kwadratowych. Liczbę w pierwszym nawiasie możemy traktować jako liczbę rzędów, a w drugim jako

liczbę kolumn macierzy (tabeli). Należy pamiętać, że podobnie jak dla tablic jednowymiarowych rozmiar tablicy musi być znany w momencie kompilacji.

```
double tab2d[1000][100];
```

Listing 3.10: Przykład definicji tablicy dwuwymiarowej

Przykład

Napisz program, który do macierzy 8x8 wpisuje wartości rzędami 0,1,2,3... Następnie wypisz na ekranie tą macierz w postaci tabeli (wykorzystaj `\t` w celu zastosowania tabulacji).

Tablice wielowymiarowe mogą też być przesyłane do funkcji. Jeśli mamy w programie tablicę wielowymiarową o nazwie `int tabl[128][234]` to, aby wysłać ją do funkcji `funk` oczekującej jako parametr tablicę wielowymiarową w programie wpisujemy `funk(tabl);`. Pytanie jak wygląda deklaracja funkcji `funk`? Otóż w deklaracji funkcji kompilator musi znać wszystkie wymiary oprócz lewego skrajnego. A więc w naszym przypadku będzie `int funk(int t[][234])`. Można oczywiście podać ten lewy skrajny indeks `int funk(int t[128][234])` i nie jest to błąd. Błędem byłoby `int funk(int t[][])`. Dla przypadku trójwymiarowego byłoby np. `int funk(int t[][100][200])`, itd.

3.6 Zadania do rozwiązania

Zadania na ocenę 3,0

Zad. 1. Napisz program, który posiada tablicę T o dwu elementach całkowitych. Program pobiera od użytkownika dwie liczby i umieszcza w tablicy, a następnie wypisuje pierwszy element tablicy T.

Zad. 2. Napisz program, który pozwala wprowadzić 8 liczb rzeczywistych i umieszcza je w tablicy. Następnie program prosi o wprowadzenie jeszcze jednej liczby L i sprawdza czy znajduje się ona w tablicy i wypisuje odpowiedni komunikat.

Zad. 3. Napisz program, który: a) pozwala wprowadzić liczbę naturalną n z zakresu $1 \div 10$, b) pozwala wprowadzić n liczb całkowitych, c) wypisuje wszystkie podane liczby, ale od końca.

Zadania na ocenę 3,5-4,0

Zad. 4. Napisz program, który pobiera od użytkownika 10 liczb rzeczywistych i wypisuje wszystkie, które są większe od ostatniej.

Zad. 5. Napisz program, który: a) pozwala wprowadzić liczbę naturalną n z

zakresu $1 \div 10$, b) pozwala wprowadzić n liczb rzeczywistych i umieszcza je w tablicy, c) oblicza wartość średnią z wprowadzonych liczb, d) wyświetla wszystkie liczby, które są większe od średniej.

Zad. 6. Napisz program, który losuje 100 liczb z przedziału $0 \div 1$. Program liczy średnią tych liczb i wypisuje, ile z tych liczb jest mniejsza niż średnia. Podpowiedź: liczby losowe możesz dodać za pomocą `#include <cstdlib>` i następnie np. `int a = rand()` [3].

Zadania na ocenę 4,5-5,0

Zad. 7. Napisz program, który: a) posiada tablicę $T1$ dla 100 liczb całkowitych z przedziału $-100 \div 100$ wypełnioną w sposób losowy, b) posiada tablicę $T2$ dla czterech całkowitych z przedziału $-100 \div 100$ i pozwala wprowadzić te liczby, c) dla każdej liczby z tablicy $T2$ oblicza się: ile razy w tablicy $T1$ występuje wartość mniejsza, d) obliczone ilości wyświetla w kolejnych wierszach.

Zad. 8. Program powinien: a) losować 20 liczb całkowitych, zapamiętać je w tablicy i pokazać użytkownikowi, b) użytkownik podaje liczbę całkowitą, c) jeśli podana liczba nie jest jedną z wylosowanych, to program kończy się komunikatem „Zle", d) w przeciwnym przypadku program wypisuje, który nr w tablicy ma wylosowana liczba.

Zad. 9. Napisz program, który: posiada dwuwymiarową tablicę o dwu wierszach i 10 kolumnach, b) pozwala wprowadzić 10 liczb rzeczywistych i umieszcza je w pierwszym wierszu tablicy, c) dla każdej wartości X z pierwszego wiersza oblicza wartość $Y = X - sin(X)$ i umieszcza w drugim wierszu tablicy, d) wypisuje w arkuszu w kolejnych wierszach kolejne wartości X i Y. Podpowiedź: funkcje matematyczne możesz dodać za pomocą `#include <math>` [2].

Zadania nieobowiązkowe

Zad. 10. Program a) losuje i zapamiętuje liczbę całkowitą z przedziału od $1 \div 100$, b) użytkownik podaje swoją liczbę, a program ją zapamiętuje, c) jeśli użytkownik podał dobrą liczbę to program pisze „OK zgadłeś za x razem" (gdzie x to nr próby) i w następnym wierszu wyświetla wszystkie liczby podane wcześniej przez użytkownika i kończy się, d) jeśli podana liczba jest za duża, program wypisuje „za duza", jeśli za mała to „za mala", e) jeśli użytkownik nie zgadł po raz 20, to program kończy się, jeśli prób było mniej, to użytkownik może podać kolejną liczbę.

Zad. 11. Napisz program, który: a) pozwala wprowadzić liczbę naturalną n z zakresu $1 \div 100$, b) w kolejnych wierszach pozwala wprowadzić n zestawów danych - po 3 liczby rzeczywiste. Każdy zestaw to: współrzędna x, współrzędna y i masa. Dane są zapamiętywane w tablicy dwuwymiarowej, c) na podstawie

danych program oblicza X i Y środka masy układu podanych n punktów materialnych:

$$X = \frac{\sum_i x_i m_i}{\sum_i m_i} \qquad Y = \frac{\sum_i y_i m_i}{\sum_i m_i} \tag{3.1}$$

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.