

Podstawy programowania: Laboratorium nr 4

Tablice dynamicznie i statyczne.

28 października 2016

mgr inż. Przemysław Walkowiak

dr inż. Michał Ciesielczyk

Instrukcja

W czasie pisania programu pamiętaj o:

1. dbaniu o czytelność kodu (odpowiednie formatowanie kodu, nazewnictwo zmiennych adekwatne do ich znaczenia, komentarze),
2. dbaniu o czytelność interfejsu z użytkownikiem (w sposób jawny pytaj użytkownika jakie dane ma podać oraz opisz wyniki, które zwracasz),
3. przed fragmentem implementującym poszczególne zadania umieść komentarz: `/*Zadanie X */` oraz wypisz na ekranie analogiczny komunikat (X jest numerem zadania): `std::cout << "Zadanie X"<< std::endl; ,`
4. umieszczeniu wszystkich rozwiązań w jednym pliku, chyba, że w poleceniu napisano inaczej.
5. w zadaniach wymagających udzielenia komentarza bądź odpowiedzi, należy umieścić go w kodzie programu (np. w postaci komentarza albo wydrukować na ekranie).

Wprowadzenie

Tablice pozwalają na przechowywanie uporządkowanych danych takiego samego typu. W tego typu kontenerach, poszczególne elementy dostępne są za pomocą indeksu. Indeks ten powinien przyjmować wartości numeryczne od 0 do $n - 1$, gdzie n jest rozmiarem tablicy.

Rozmiar tablicy jest albo ustalony z góry (tablice statyczne), albo może się zmieniać w trakcie wykonywania programu (tablice dynamiczne).

Tablice statyczne – `std::array`

Odpowiednikiem tablic statycznych w C++ jest `std::array`. Rozmiar tablicy statycznej musi być znany w trakcie kompilacji. Aby móc skorzystać z tablic `std::array` w C++ należy załączyć bibliotekę `<array>`, tj. dodać do programu dyrektywę: `#include <array>`.

```
#include <array>
using namespace std;

/* ... */

5 // inicjalizacja 3-elementowej tablicy liczb
  // całkowitych wartościami [1,2,3]
  array<int, 3> arr {1,2,3};

10 // inicjalizacja 5-elementowej tablicy liczb zmiennoprzecinkowych
   array<float, 5> arr2;
```

```
// pobieranie i-tego elementu z tablicy (licząc od '0')
int i = 1;
15 int x = arr[i]; // x = 2

// odczytywanie wielkości kolekcji (liczby elementów)
unsigned int n = arr.size(); // n = 3

20 // ustawianie wartości i-tego elementu tablicy
arr[i] = -1;
```

Tablice dynamiczne – `std::vector`

Odpowiednikiem tablic dynamicznych w C++ jest `std::vector`. Aby móc skorzystać z tablic dynamicznych w C++ należy załączyć bibliotekę `<vector>`.

```
#include <vector>
using namespace std;

/* ... */
5 // inicjalizacja pustego wektora liczb całkowitych
vector<int> v;

// inicjalizacja wektora liczb całkowitych o rozmiarze 10
10 auto v2 = vector<int>(10);
vector<int> v3(10);

// dodawanie nowej wartości na końcu wektora
v.push_back(8);
15 int x = 2;
v.push_back(x);

// pobieranie i-tego elementu z wektora (licząc od '0')
int y = v[i];
20 // odczytywanie wielkości kolekcji (liczby elementów)
unsigned int n = v.size()

// ustawianie wartości i-tego elementu tablicy
25 v[i] = -1;

// usuwanie i-tego elementu (licząc od '0')
v.erase(v.begin() + i);

30 // czyszczenie wektora (usuwanie wszystkich elementów)
v.clear();
```

```
35 // zmiana/ustalenie rozmiaru wektora
// oraz wypełnienie wartościami domyślnymi nowych elementów
v.resize(20);

// zmiana/ustalenie rozmiaru wektora
// oraz wypełnienie wartościami 1337 nowych elementów
v.resize(20, 1337);
```

Pętla range-based for

Pętla range-based for pozwala iterować po wszystkich elementach kolekcji. Przykładowo:

```
5 std::vector<float> v;
// ...
for (auto e : v) {
    cout << e; // wyświetla kolejne elementy kolekcji
}

std::array<int, 3> a {1,2,3};
// ...
10 for (auto e : a) {
    cout << e; // wyświetla kolejne elementy kolekcji
}
```

Specyfikator `auto` pozwala na automatyczne wnioskowanie typu zmiennej na podstawie sposobu w jaki jest inicjalizowana. Jednakże zmienna nadal ma jeden, stały typ przez cały czas działania programu!

Zadania

Zadanie 1

Zdefiniuj nową tablicę statyczną liczb całkowitych o rozmiarze 100. Umieść w niej kolejno liczby całkowite z przedziału $\langle 100, 199 \rangle$. Następnie, oblicz sumę wszystkich elementów w tablicy. Wyświetl na ekranie wynik działania programu.

Wskazówka Do sumowania wszystkich elementów tablicy możesz wykorzystać pętlę range-based for.

Zadanie 2

Zdefiniuj funkcję `void print(vector<int> v)` wypisującą na ekranie całą zawartość wektora liczb całkowitych w następujący sposób: $[v_0, v_1, \dots, v_{n-1}]$, gdzie v_i to kolejne elementy wektora o rozmiarze n . Następnie, przetestuj swoją implementację wyświetlając zawartość przykładowego wektora, np.:

```
vector<int> v{ 1,2,3,4 };  
print(v); // powinno wyswietlic na ekranie: [1,2,3,4]
```

Zadanie 3

Zdefiniuj następujące funkcje:

- a) `float sum(vector<float> v)` – zwracającą sumę wszystkich elementów wektora,
- b) `float average(vector<float> v)` – zwracającą średnią wszystkich elementów wektora,
- c) `array<float, 2> minmax(vector<float> v)` – zwracającą tablicę, w której pierwszy element to minimum podanego wektora, a drugi element to jego maksimum.

Następnie, wczytaj od użytkownika n liczb zmiennoprzecinkowych (wartość n również wczytaj od użytkownika). Wykorzystując zaimplementowane przez Ciebie funkcje wyświetl statystyki wczytanego wektora (sumę, średnią, minimum oraz maksimum).

Zadanie 4

Zdefiniuj funkcję `vector<int> randomVector(unsigned int size, int min, int max)` zwracającą nowy wektor o rozmiarze `size`, którego elementy zostały zainicjalizowane liczbami losowymi z przedziału od `min` do `max`. Wykorzystaj funkcję z poprzedniego zadania do wyświetlenia przykładowego wyniku działania funkcji `randomVector`.

Wskazówka Do generowania liczb losowych możesz wykorzystać następującą funkcję:

```
#include <random>  
  
int randomInt(int min, int max) {  
    static default_random_engine e{};  
    uniform_int_distribution<int> d(min, max);  
    return d(e);  
}
```

Zadanie 5

Zainicjalizuj dwa wektory o długości n wartościami losowymi z przedziału $\langle 3; 27 \rangle$. Oblicz ich iloczyn skalarny oraz wyświetl całe działanie na ekranie. Wartość n wczytaj od użytkownika. Wykorzystaj funkcje zdefiniowane w poprzednich zadaniach.

Wskazówka 1 iloczyn skalarny:

$$a \cdot b = \sum_{i=1}^n a_i b_i, \quad (1)$$

gdzie $a, b \in \mathbb{R}^n, n = 1, 2, \dots$,

Zadanie 6

Wczytaj od użytkownika dowolny napis. Następnie, wykonaj na nim poniższe operacje:

- zamień wszystkie litery na wielkie,
- zamień wszystkie litery na małe,
- zamień wielkie litery na małe i małe na wielkie.

Przyjmij, że podane zostały wyłącznie znaki z tablicy ASCII.

Dodatkowe informacje:

- Tablica ASCII: <http://cplusplus.com/doc/ascii/>.
- Sprawdzanie długości napisu: `string::size`.

Wskazówka 1 Do wczytywania napisów wykorzystaj funkcję `std::getline`, np.:

```
string napis;  
getline(cin, napis);
```

Wskazówka 2 Przejrzyj jakie użyteczne funkcje znajdują się w bibliotece standardowej: <http://en.cppreference.com/w/cpp/string/byte>. Wykorzystaj je do sprawdzenia czy dany znak jest literą oraz do sprawdzenia/zmiany jej wielkości.

Zadanie 7

Wczytaj od użytkownika napis o długości do 30 znaków, a następnie wyświetl go od końca (napisz funkcję, która odwraca napis). Sprawdź czy podany napis jest palindromem oraz określ jego parzystość/nieparzystość. Przy sprawdzaniu należy zignorować znaki białe.

Palindrom ciąg znaków, który czytany od początku i od końca ma taką samą postać i znaczenie. Np. *anna* (palindrom parzysty), *kajak* (palindrom nieparzysty), *'Nogawka jak wagon'*.

Zadanie 8

Utwórz dwie macierze losowe A i B o wartościach z przedziału $\langle -16; -4 \rangle$ i wymiarach $m \times n$, a następnie wykonaj operacje $A + B$ i $A - B$. Wartość m, n wczytaj od użytkownika.

Wskazówka Macierz można zaprezentować jako wektor wektorów: `vector<vector<int>>`. Pamiętaj o odpowiednim zainicjalizowaniu każdego wektora w takiej macierzy. Możesz wykorzystać funkcję `resize` wektora.

Zadanie 9

Utwórz dwie macierze losowe A i B o wartościach z przedziału $\langle 1; 14 \rangle$ i wymiarach odpowiednio $m \times n$ oraz $n \times p$, a następnie wykonaj operację $A * B$. Wartość m, n, p wczytaj od użytkownika.

Dodatkowe informacje:

- Mnożenie macierzy – http://pl.wikipedia.org/wiki/Mno%C5%BCenie_macierzy

Na następne zajęcia

- Specyfikator `const` – <http://en.cppreference.com/w/cpp/language/cv>.
- Przekazywanie argumentów przez referencję: <http://www.cplusplus.com/doc/tutorial1/functions/#reference>.
- Stałe referencje: <http://www.cplusplus.com/doc/tutorial/functions/#constref>.
- Obsługa plików tekstowych: `std::fstream`, `std::ofstream` – <http://www.cplusplus.com/reference/iostream/>.