Trudne się wylosowało, czyli tablice

W ramach wcześniejszych zajęć pisaliśmy programy, które czasem wymagały od użytkownika podania jakichś danych, na przykład liczby, albo dwóch. To było dosyć proste:

```
int a, b;
cout<<"podaj liczby a i b:";
cin>>a>>b;
```

Co jednak w wypadku, kiedy będziemy musieli podać tysiąc liczb?

```
int a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z;
cout<<"podaj liczby a i b:";
cin>>a>>b>>c>>d>>e>>f>>g>>h>>i>>j>>k>>l>>m>>n>>o>>p>>q>>r>>s>>t>>u>>v>>w>>x>>y>>z;
```

Tu jest 26, a już rozwiązanie jest bardzo nieeleganckie i przy okazji dość głupie. Jeśli mamy dużą ilość danych tego samego typu do przetworzenia, to powinniśmy zastosować mechanizm tablic. Tablice są podstawowym elementem służącym organizacji danych w programowaniu. Tablica jest de facto po prostu zwykłą zmienną, tylko przechowującą wiele wartości tego samego typu. Dla zobrazowania: załóżmy, że mamy do przechowania 17 wartości ocen studentów:

```
double grades[17];
```

Powyższa deklaracja utworzy nam tablicę, która będzie w stanie przechowywać 17 wartości typu double.

- `Double` – typ używanych danych
- `Grades` – nazwa tablicy
- `[17]` – rozmiar tablicy

Powyższa deklaracja utworzy nam tablicę pustą (ewentualnie wypełnioną siedemnastoma losowymi wartościami). Tutaj ważna uwaga – tworząc tablice w ten sposób nie możemy potem zmienić jej typu ani rozmiaru – musi on być znany w momencie kompilacji, nie może być podany przez użytkownika. Można tworzyć tablice dynamiczne, ale to sobie zostawimy do momentu poznania wskaźników.

```
int example[]={21, 37, 4, 20, 6, 9};
```

Tablice można również inicjalizować podając im od razu dane, wówczas nie musimy podawać rozmiaru – kompilator sam go sobie obliczy na podstawie podanych danych.

```
double grades[6] = {2, 4, 5};
```

Taki sposób również jest dopuszczalny – w tym wypadku ostatnie 3 komórki w tablicy będą puste/zawierać jakieś śmieci nad którymi nie mamy kontroli.

Dostęp do elementów tablicy jest równie prosty, co inicjalizacja. Na przykład z tablicy grades, zawierającej elementy:

WARTOŚĆ	2	4.5	4	3	4.5	3.5	3	2	5	4.5
INDEKS	0	1	2	3	4	5	6	7	8	9

Chcielibyśmy znaleźć wartość znajdującą się na pozycji 5.

```
double grades[10] = {2, 4.5, 4, 3, 4.5, 3.5, 3, 2, 5, 4.5};
cout<<"Ocena z indeksu 5: "<<grades[5];
```

Taki programik wydrukuje nam wartość 3.5. Warto w tym miejscu zwrócić uwagę, że pierwszy element tablicy (2) ma indeks 0. Indeksy zaczynają się zawsze od 0, a kończą na wartości rozmiar_tablicy – 1. Rozmiar tablicy możemy poznać stosując operator sizeof(). Zwróci on nam rozmiar argumentu w bajtach, dlatego

```
sizeof(grades)
```

w przypadku powyższej tablicy zwróci 80. Aby szybko ustalić długość tablicy (liczbę jej elementów) musimy podzielić wartość zwróconą przez ty danych tej tablicy:

```
sizeof(grades)/sizeof(double)
```

Wypełnianie tablic

Oczywiście dopuszczalnym jest zapisywanie danych do tablicy w dokładnie taki sam sposób, jak do zwykłej zmiennej – pamiętać jednak musimy, że zwykle zapisujemy (i odczytujemy) te dane seryjnie. Można zatem tak:

```
cin>>grades[0]>>grades[1];
```

Ale nie ma tu żadnej elastyczności, napisanie kodu dla przyjmowania 100 wartości dla tablicy byłoby czasochłonne i niepotrzebne, wystarczy bowiem zastosować pętlę for, aby wszystko stało się dużo przyjemniejsze.

```
double grades[10];
    for (int i=0; i<10; i++){
        cout<<"Podaj ocenę "<<i+1<<": ";
        cin>>grades[i];
    }
```

Dokładnie tak samo będzie działał wydruk pojedynczych elementów. Należy jednak pilnować, żeby nie wywoływać elementów spoza tablicy! Jeśli tablica ma 10 elementów, to tablica[10] nie istnieje!

Zadania

1. Napisz program, liczący ciąg Fibonacciego. Program powinien przyjmować od użytkownika, ile elementów ciągu chce uzyskać, a następnie wydrukować wszystkie na terminalu

2. Napisz program, który wypełni losowymi liczbami o zakresie podanym przez użytkownika tablicę o rozmiarze 100, a następnie obliczy średnią wartość w tablicy i wydrukuje oddzielnie wszystkie elementy większe i mniejsze od tej średniej

```
rand()%max;
```