Sformułowanie problemu wyszukiwania

W programach wyszukujących określony element w zbiorze danymi wejściowymi będzie ciąg *n* losowych liczb całkowitych: $a_0, a_1, \ldots, a_{n-1}$. Liczby zapamiętamy w **tablicy**.

Funkcja główna w programach, które utworzymy (niezależnie od zastosowanej metody wyszukiwania) powinna uwzględniać następujące etapy:

1. Wylosowanie tablicy liczb
2. Wypisanie tablicy liczb
3. Wyszukanie elementu w tablicy
4. Wypisanie wyniku wyszukania

Aby zrealizować dwa pierwsze etapy zdefiniujemy funkcje `Losuj` i `Wypisz`. Liczba elementów tablicy, czyli rozmiar tablicy, powinna być znana w momencie kompilacji programu. Dlatego rozmiar określimy za pomocą **stałej**, którą zdefiniujemy przed funkcjami. Jeśli chcielibyśmy sprawdzić działanie programu dla tablicy o innej liczbie elementów, wystarczy wówczas wprowadzić zmianę tylko w jednej linii kodu – w definicji stałej podać inny rozmiar tablicy.

Uwaga: Powyższa uwaga dotycząca określania rozmiaru tablicy dotyczy standardu języka C++, a więc zachowując te zasady programy będą działały we wszystkich środowiskach i kompilatorach języka C++. CodeBlocks i Dev dopuszczają wczytanie rozmiaru tablicy, ale nie można wtedy później zmienić rozmiaru tablicy.

```
int n;
cout<<"podaj rozmiar tablicy: "; cin>>n;
int tab[n];
```

Fragment kodu źródłowego programu wyszukującego element tablicy z definicjami stałej oraz funkcji losującej i wypisującej tablicę liczb może być następujący:

```
 1 #include <iostream>
 2 #include <cstdlib>
 3 #include <ctime>
 4
 5 using namespace std;
 6
 7 const int N=10;
 8
 9 void Losuj(int A[])
10 {
11     for (int i=0;i<N;i++) A[i]=rand()%100;
12 }
13
14 void Wypisz(int A[])
15 {
16     for (int i=0;i<N;i++) cout<<A[i]<<" ";
17     cout<<endl;
18 }
```

W linii 7 została zdefiniowana stała `N` o wartości 10 określająca rozmiar tablicy. Parametrami funkcji `Losuj` i `Wypisz` jest tablica. W linii 11 tablica wypełniana jest losowymi liczbami z zakresu od 0 do 99. Zakres losowania nie ma znaczenia. Został ograniczony, żeby łatwo było zinterpretować działanie programu. Otrzymanie losowej liczby całkowitej nieujemnej umożliwia **funkcja** `rand`. Aby móc skorzystać z tej funkcji, konieczne jest dołączenie do programu biblioteki `cstdlib`. Zakres losowania ograniczamy biorąc resztę z dzielenia wylosowanej liczby przez 100.

Przypomnijmy, że **przekazanie parametru przez wartość** oznacza, że parametr funkcji jest chroniony, tzn. wszystkie operacje są wykonywane na kopii parametru, traktowanego jak **zmienna lokalna**. W przypadku, gdy parametrem funkcji jest tablica, w rzeczywistości parametrem funkcji nie jest tablica, a adres tej tablicy – tablice w języku C++ są reprezentowane za pośrednictwem adresu. Oznacza to, że wszystkie operacje są wykonywane na oryginalnej tablicy, a dokonane w niej zmiany zapamiętane po zakończeniu działania funkcji. Taki sposób przekazania parametru nazywamy **przekazaniem parametru przez wskaźnik**. Tak więc wartości zwracane przez funkcję `Losuj` są zachowane po zakończeniu jej działania.

Fragment kodu źródłowego funkcji `main` wywołującej funkcje `Losuj` i `Wypisz` może być następujący:

```
1 int A[N];
2 srand(time(NULL));
3 Losuj(A);
4 Wypisz(A);
```

W linii 1 zadeklarowana jest tablica `N` liczb całkowitych. Instrukcja w linii 2 inicjalizuje generator liczb losowych i uzależnia losowanie liczb od zegara komputera. Jeśli nie dodalibyśmy tej instrukcji, po każdym uruchomieniu programu wynikiem funkcji `Losuj` byłby ten sam ciąg liczb. Ustawienie punktu startowego dla mechanizmu losującego umożliwia **funkcja** `srand`. Z kolei dzięki **funkcji** `time` zwrócimy aktualny czas w postaci liczby całkowitej. Korzystanie z funkcji `srand` wymaga dołączenia biblioteki `cstdlib`, a korzystanie z funkcji `time` – biblioteki `ctime`.

Poszukujemy liczby w zbiorze nieuporządkowanym

Jeżeli poszukujemy danego elementu w zbiorze, o którym nie wiemy czy jest uporządkowany, pozostaje nam sprawdzać, czy pierwszy element jest równy poszukiwanemu, jeśli nie, czy drugi element jest równy poszukiwanemu itd. Czynności powtarzamy dopóki nie odnajdziemy elementu lub gdy przejrzymy wszystkie elementy zbioru. Taki sposób postępowania nazywamy **przeszukiwaniem liniowym** lub **wyszukiwaniem sekwencyjnym.**

Oto specyfikacja problemu wyszukiwania elementu w zbiorze nieporządkowanym:

Specyfikacja

Dane: `A[0..n-1]` – tablica *n* losowych liczb całkowitych,
x – liczba całkowita.

Wynik: wartość `true`, gdy liczba *x* znajduje się w tablicy, `false` – w przeciwnym razie.

Możemy mieć szczęście i znaleźć szukany element przy pierwszym porównaniu. Jeśli elementu nie będzie w tablicy lub znajdzie się na ostatnim miejscu, to musimy przejrzeć całą tablicę, czyli wykonać *n* porównań. Średnio wykonamy około $\frac{n}{2}$ porównań.

Oto przykładowy zapis w pseudokodzie funkcji sprawdzającej, czy liczba *x* znajduje się w *n*-elementowej tablicy `A`:

```
funkcja SzukajLin(A[],x)
    dla i ← 0, 1,..., n - 1 wykonuj
        jeśli A[i] = x to zwróć prawda i zakończ
    zwróć fałsz i zakończ
```

Kod funkcji w języku C++ i program testujący mogą być następujące:

```
20  bool SzukajLin (int A[], int x)
21  {
22      for (int i=0;i<N;i++)
23          if (A[i]==x) return true;
24      return false;
25  }
26
27  int main ()
28  {
29      int x;
30      int A[N];
31      srand(time(NULL));
32      Losuj(A);
33      Wypisz(A);
34      cout<<"Podaj liczbe do wyszukania: "; cin>>x;
35      if (SzukajLin(A,x)) cout<<"Jest";
36      else cout<<"Nie ma";
37      return 0;
38  }
```

Zadania:

1. Napisz program losujący, wypisujący tablicę liczb, następnie wczytujący liczbę i sprawdzający metodą przeszukiwania liniowego, czy liczba występuje w tablicy.
2. Napisz program losujący, wypisujący tablicę liczb, następnie znajdujący w niej minimum i maksimum metodą przeszukiwania liniowego.
3. Napisz program symulujący n rzutów kostką sześcienną do gry i zliczający, ile razy wystąpiły poszczególne liczby oczek.