Практична робота

Тема: Пошук у масивах: послідовний пошук і бінарний пошук.

Мета: Навчитися здійснювати послідовний і бінарний пошук у масивах.

*Короткі теоретичні відомості*

Часто програмісту доводиться працювати з великим об’ємом даних, який зберігається у вигляді масиву. Може виникнути потреба вияснити, чи є певний елемент у масиві. Процес знаходження якогось елементу масиву називають пошуком. Розрізняють два способи пошуку – простий послідовний пошук і більш ефективний бінарний пошук (дихотомія).

Послідовний пошук порівнює кожний елемент масиву з ключем пошуку. Оскільки масив не впорядкований, цілком ймовірно, що шукане значення виявиться першим елементом масиву. Але в середньому програма повинна порівнювати з ключем пошуку половину елементів масиві.

// *Послідовний пошук*

//#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

using namespace std;

int linearSearch(const int [],int, int);

int main()

{

srand(time(0));

const int arraySize=100;

int a[arraySize], searchKey, element;

for (int x=0; x<arraySize; x++)

{

a[x]=1+rand()%100;

if (x%10==0)

cout<<endl;

cout<<setw(4)<<a[x];

}

cout<<"\nEnter the key\n\n";

cin>>searchKey;

element=linearSearch (a, searchKey,arraySize);

if (element!=-1)

cout<<"Element = "<<element<<"\n\n";

else

cout<<"Element NOT"<<"\n\n";

return 0;

}

int linearSearch(const int a[], int searchKey, int arraySize)

{

for (int n=0; n<arraySize;n++)

if (a[n]==searchKey)

{

cout<<"a["<<n<<"]="<<a[n]<<"\n\n";

return n;

}

return -1;

}

Метод послідовного пошуку добре працює для невеликих або не відсортованих масивів. Для великих масивів послідовний пошук малоефективний. Якщо масив відсортований можна використати метод бінарного пошуку.

Алгоритм бінарного пошуку виключає половину ще неперевірених елементів масиву після кожного порівняння. Алгоритм визначає місцеположення середнього елементу масиву і порівнює його з ключем пошуку. Якщо вони дорівнюють, то ключ пошуку знайдено і видається індекс цього елементу. В противному випадку задача скорочується на половину елементів масиву. Якщо ключ пошуку менше, чим середній елемент масиву, то подальший пошук здійснюється в першій половині масиву, а якщо – більше, то- в другій половині. Якщо ключ пошуку не співпадає з середнім елементом вибраного підмасиву (частини вихідного масиву), то алгоритм повторно застосовується і скорочує область пошуку до четверті вихідного масиву. Пошук продовжується до тих пір, поки ключ пошуку не стане рівним середньому елементу, або поки остаточний підмасив містить хоча б один елемент, не рівний ключу пошуку, тобто поки не знайдено ключ пошуку.

// *Бінарний пошук у масивах*

#include <iostream>

#include <iomanip>

using namespace std;

int binarySearch(const int [], int, int, int,int);

void printHeader(int);

void printRow(const int [], int, int, int, int);

int main()

{

const int arraySize=10;

int a[arraySize], key, result;

for (int i=0; i<arraySize;i++)

a[i]=2\*i;

cout<<"Enter the number: 0..28:";

cin>>key;

printHeader(arraySize);

result=binarySearch(a,key,0,arraySize-1,arraySize);

if (result!=-1)

cout<<"\n"<<key<<" "<<result<<endl;

else

cout<<"\n"<<key<<" NOT!!!"<<endl;

return 0;

}

//

int binarySearch(const int b[], int searchKey, int low, int high, int size)

{

int middle;// середина

while (low<=high)

{

middle=(low+high)/2;

printRow(b, low, middle, high, size);

if (searchKey==b[middle])

return middle;

else

if(searchKey<b[middle])

high=middle-1;

else

low=middle+1;

}

return -1;

}

//

void printHeader(int size)

{

int i;

cout<<"Index\n";

for (i=0; i<size; i++)

cout<<setw(3)<<i<<" ";

cout<<"\n";

for (i=1; i<=4\*size; i++)

cout<<"-";

cout<<endl;

}

//

//

void printRow(const int b[], int low, int mid, int high, int size)

{

for (int i=0; i<size;i++)

if (i<low || i>high)

cout<<" ";

else

if(i==mid)

cout<<setw(3)<<b[i]<<"\*";

else

cout<<setw(3)<<b[i]<<" ";

cout<<endl;

}

*Передача масивів у функції.*

Щоб передати масив в якості аргументу в функцію, потрібно вказати тільки ім’я масиву. Наприклад для масиву *int* ***a****[arraySize]:*

element=linearSearch (**a**, searchKey,arraySize);

При передачі масиву у функцію його розмір потрібно передавати для того, щоб функція могла опрацювати задане число елементів масиву.

Можлива ситуація, коли функції неможна дозволяти модифікувати (змінювати елементи масиву). У С++ є специфікатор типу **const** для заборони модифікації значень масиву у функції.

int linearSearch(const int a[], int searchKey, int arraySize)

Коли перед параметром стоїть специфікатор const, то елементи масиву в тілі функції стають постійними і будь-яка спроба змінити елементи масиву у тілі функції призводить до помилки при трансляції