Практическое занятие № 3 «Программирование алгоритмов поиска в одномерных массивах на языке С++»

Учебные цели:

- получение умений и навыков создания и отладки программ для обработки одномерных массивов;
- исследовать методы последовательного и логарифмического (бинарного) поиска информации в массиве;
- получение умений и навыков создания и отладки программ с циклической структурой.

Воспитательные цели:

- формировать диалектико-материалистическое мировоззрение;
- формировать навыки самостоятельности и дисциплинированности;
- стимулировать активную познавательную деятельность обучаемых, способствовать формированию у них творческого мышления.

Категория слушателей: 2,3 курс.

Время: 90 мин.

Место проведения: компьютерный класс.

Материально-техническое обеспечение:

1) персональный компьютер *IBM PC* с операционной системой Windows XP; 2) среда разработки приложений *Visual C++.NET.*

ПЛАН ПРАКТИЧЕСКОГО ЗАНЯТИЯ

Учебные вопросы	Время, мин
Вступительная часть	5
1. Поиск заданных элементов одномерного массива в С++. .	15
2. Выполнение индивидуального задания	65
Заключительная часть	5

Элементы теории

Для нахождения информации в неупорядоченном массиве требуется последовательный поиск, начинающийся с первого элемента и заканчивающийся при обнаружении подходящих данных либо при достижении конца массива. Этот метод применим для неупорядоченной информации, но также можно использовать его и на отсортированных

данных. Однако если данные уже отсортированы, можно применить двоичный поиск, который находит данные быстрее.

Последовательный поиск очень легко запрограммировать. Приведенная ниже функция осуществляет поиск в массиве символов известной длины, пока не будет найден элемент с заданным ключом:

```
/* Последовательный поиск */
...
int i;
for(i=0; i < n; i++)
   if (b == A[i]) {cout << i; break;}
if (i>=n) cout << "элемент не найден";
```

Здесь A − указатель на массив, содержащий информацию. Функция возвращает индекс подходящего элемента, если таковой существует, либо -1 в противном случае.

Понятно, что последовательный поиск в среднем просматривает n/2 элементов. В лучшем случае он проверяет только один элемент, а в худшем − n. Если информация хранится на диске, поиск может занимать продолжительное время. Но если данные не упорядочены, последовательный поиск − единственно возможный метод.

В связи с малой эффективностью по сравнению с другими алгоритмами линейный поиск обычно используют только если отрезок поиска содержит очень мало элементов, тем не менее, линейный поиск не требует дополнительной памяти или обработки/анализа функции, так что может работать в потоковом режиме при непосредственном получении данных из любого источника. Так же, линейный поиск часто используется в виде линейных алгоритмов поиска максимума/минимума.

Двоичный поиск. Если данные, в которых производится поиск, отсортированы, для нахождения элемента можно применять метод, намного превосходящий предыдущий − двоичный поиск. Есть и другие названия: *дихотомический поиск*, *логарифмический поиск*, *поиск делением пополам*. В нем применяется метод половинного деления. Сначала проверим средний элемент. Если он больше, чем искомый ключ, проверим средний элемент первой половины, в противном случае − средний элемент второй половины. Будем повторять эту процедуру до тех пор, пока искомый элемент не будет найден либо пока не останется очередного элемента.

Например, чтобы найти число 4 в массиве A={1 2 3 4 5 6 7 8 9} при двоичном поиске сначала проверяется средний элемент − число 5. Поскольку оно больше, чем 4, поиск продолжается в первой половине:

{1 2 3 4}.

Средний элемент теперь равен 2. Это меньше, чем 4, поэтому первая половина отбрасывается. Поиск продолжается в части {3 4}.

Средний элемент теперь равен 3. Это меньше, чем 4, поэтому первая половина отбрасывается (элемент {3}). Поиск продолжается в части {4}.

Средний элемент теперь равен 4. На этот раз искомый элемент найден.

В двоичном поиске количество сравнений в худшем случае равно $\log_2(n)$.

В среднем случае количество немного ниже, а в лучшем − количество сравнений равно 1.

Ниже приведена функция двоичного поиска для массива целых чисел.

```
/* Двоичный поиск */
...
int low, high, mid;
low = 0; high = n-1;
while(low <= high)
{
mid = (low+high)/2;
if(b < A[mid]) high = mid-1;
   else
       if(b > A[mid]) low = mid+1;
          else {cout << mid; break;} /* число найдено */
}
if (low > high) cout << "элемент не найден";
```

Практические приложения метода двоичного поиска разнообразны:

- Широкое распространение в информатике применительно к поиску в структурах данных. Например, поиск в массивах данных осуществляется по ключу, присвоенному каждому из элементов массива (в простейшем случае сам элемент является ключом).
- Также его применяют в качестве численного метода для нахождения приближённого решения уравнений.

Вопросы для самопроверки

1. Сформулируйте идею метода последовательного поиска.
2. Сформулируйте идею бинарного поиска.
3. В каких случаях применяют методы последовательного и бинарного поиска?
4. Какой массив называется упорядоченным?
5. Напишите алгоритм поиска максимального элемента в одномерном массиве.
6. Напишите алгоритм поиска минимального элемента в одномерном массиве.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОТРАБОТКЕ УЧЕБНЫХ ВОПРОСОВ

Практическое задание

Задание 1. В соответствии с Вашим вариантом составьте и отладьте программу обработки одномерного массива.

1. Задан одномерный массив целых чисел. Напишите программу, которая определит, сколько элементов массива входит в диапазон (a, b), где a и b – числа, которые пользователь вводит с клавиатуры. В случае, если такого диапазона чисел в массиве нет – вывести соответствующее сообщение.
2. Напишите программу, которая определяет количество элементов массива равных введенному элементу с клавиатуры.
3. Найти максимальный элемент одномерного массива, предварительно отсортировав его элементы по возрастанию. В качестве алгоритма сортировки выбрать любой: методом обмена, методом вставки, методом выбора.
4. Напишите программу, которая определяет количество элементов массива кратных введенному элементу с клавиатуры.
5. Найти минимальный элемент одномерного массива, предварительно отсортировав его элементы по убыванию. В качестве алгоритма сортировки выбрать любой: методом обмена, методом вставки, методом выбора.
6. Дан одномерный массив размерности n, отсортированный по возрастанию. Найти место в массиве, на котором следует расположить элемент m, чтобы не нарушить возрастающую последовательность чисел. Значение элемента m вводится с клавиатуры.
7. Дан одномерный массив размерности n, отсортированный по убыванию. Найти место в массиве, на котором следует расположить элемент m, чтобы не нарушить убывающую последовательность чисел. Значение элемента m вводится с клавиатуры.
8. Дан одномерный массив размерности n. Напишите программу, которая определяет сумму элементов, которые находятся правее элемента массива введенного пользователем с клавиатуры. В случае, если такого элемента в массиве нет – вывести соответствующее сообщение.
9. Задан одномерный массив целых чисел. Напишите программу, которая определит, сколько элементов массива больше введенного с клавиатуры. В случае, если таких элементов в массиве нет – вывести соответствующее сообщение.
10. Задан одномерный массив целых чисел. Напишите программу, которая определит, сколько элементов массива меньше введенного с клавиатуры. В случае, если таких элементов в массиве нет – вывести соответствующее сообщение.

Контрольные вопросы

1. Задан одномерный массив целых чисел A={1, 7, 0, -4, 6, 12}. Необходимо определить содержится ли в нем число 6. Каким методом поиска следует воспользоваться и сколько сравнений будет в нем выполнено для поиска заданного числа?
2. Задан одномерный массив целых чисел A={-5, 4, 8, 15, 19, 24, 29}. Необходимо определить содержится ли в нем число 24. Каким методом поиска следует воспользоваться и сколько сравнений будет в нем выполнено для поиска заданного числа?
3. Как в бинарном методе поиска определить, что искомый элемент не содержится в массиве?
4. Как в методе последовательного поиска определить, что искомый элемент не содержится в массиве?