**теоретические сведения**

Одна из распространенных задач при программировании задач обработки данных, работы с матричным представлением числовых данных и решения математических задач численными методами является ***сортировка массивов данных***. Зачастую скорость работы программы определяется эффективностью используемого алгоритма поиска. Часто проблему поиска в массиве записей нужных данных можно свести к задаче поиска в массиве ключевых слов.

В отличие от методов сортировки классификация методов поиска не отличается особым разнообразием. Методы поиска будут различными для упорядоченных и неупорядоченных массивов. Неупорядоченными массивами считаются массивы, о порядке следования элементов которых заранее ничего не известно. Для таких массивов особых способов поиска нет, кроме как последовательно просматривать все их элементы. В теории такой поиск называется ***линейным***. В случае, если элементы массивов каким-то образом отсортированы, то такие массивы называют упорядоченными и для поиска в них используют так называемые методы поиска в строке.

**ЛИНЕЙНЫЙ ПОИСК (Неупорядоченный поиск).** Для выполнения линейного поиска нет определенных методов. Обычно реализуют простой перебор элементов массива до тех пор, пока не будет найден нужный элемент или не будут перебраны все элементы. (Пример программы, осуществляющей линейный поиск и подсчет найденных элементов, см. ниже.)

***Программа 1.***

*Вход:* mas[n] - неупорядоченная последовательность двоичных значений длиной n, k - искомое значение.

*Выход:* i - позиция в mas[n] (0<i<n-1), соответствующая найденному символу.

*.model small*

***.data***

*result db 00h ; начальное значение результата поиска*

*;задаем массив*

*mas db 50h,08h,52h,06h,90h,17h,89h,27h,65h,42h,15h,51h,*

*db 61h,67h,76h,70h*

*n=$-mas*

*k db 15h*

***.stack 256***

***.486***

***.code***

*main:*

*mov dx,@data*

*mov ds,dx*

*;i:=0*

*xor si,si ;i=(si):=0*

*mov al,k*

*;ECЛИ k=mas[i] ТО ПЕРЕЙТИ\_НА \_exit*

*s2: cmp al,mas[si]*

*je ok*

*;i:=i+1*

*inc si*

*;ECЛИ i=<n ТО ПЕРЕЙТИ\_НА s2*

*cmp si,n-1*

*jbe s2*

*;реакция на неудачный результат поиска*

*mov result,00*

*jmp exit*

*;\_exit:*

*;реакция на удачный результат поиска*

*ok: mov result,n+1*

*jmp exit*

*exit:*

*mov ax,4c00h*

*int 21h*

*end main*

Команда xor приемник, источник является операцией логического исключающего ИЛИ. Производит операцию логического исключающего ИЛИ над парами битов двух операндов: бит результата равен 1 если значения соответствующих битов операндов различны, в остальных случаях он нулевой. Результат помещается в операнд приемник. В нашем случае операция xor si,si используется для быстрого обнуления индекса si.

**ПОИСК В УПОРЯДОЧЕННОЙ ПОСЛЕДОВАТЕЛЬНОСТИ**

В случае, если элементы массива предварительно упорядочены, то можно значительно ускорить процедуру поиска использованием так называемого алгоритма двоичного (бинарного) поиска. Он заключается в следующем.

Упорядоченный массив чисел можно представить в виде двоичного дерева. Для этого необходимо, чтобы программа выполняла следующие действия. Сначала определяется элемент, расположенный в середине исходного массива, - он будет вершиной двоичного дерева. Далее в стеке запоминаются интервалы ключей подмассивов, расположенных слева и справа от элемента – вершины двоичного дерева. После этого в цикле происходит извлечение интервалов ключей подмассивов из стека, определение серединных элементов в этих подмассивах, разбиение на очередные подмассивы, запоминание их границ в стеке, извлечение и т.д. В результате будет построено двоичное дерево.

Задачу поиска ключа с определенным значением «k» можно сравнить с обходом двоичного дерева начиная с его вершины. Если «k» меньше значения ключа вершины, то идем к узлу дерева по левой ветке, если больше – то по правой. Значение ключа в очередном узле соответствует среднему элементу подмассива, лежащему слева (или справа) от элемента, соответствующего вершине дерева. Для среднего элемента подмассива процедура сравнения и принятия решения о переходе повторяется. Процесс заканчивается, когда обнаружен узел, ключ которого равен «к», либо очередной узел является конечным и двигаться больше некуда.

***Программа 2.***

*Вход:* mas[n] - упорядоченная последовательность двоичных значений длиной N, k - искомое значение.

*Выход:* i - позиция в mas[n] (0<i<n-1), соответствующая найденному символу.

***.model small***

***.data***

*result db 0 ; начальное значение номера позиции результата поиска*

*;задаем массив*

*mas db 02h,04h,06h,08h,16h,24h,38h,45h,47h,48h,57h, 56h,58h,70h, 76h, 79h*

*n=$-mas*

*k db 4 ;искомое значение*

***.stack 256***

***.486***

***.code***

*main: mov dx,@data*

*mov ds,dx*

*;в si и di индексы первого и последнего элементов последней просмотренной ;части последовательности:*

*mov si,0*

*mov di,n-1*

*xor ax,ax*

*mov al,k ;искомое значение в ax*

*;получим центральный индекс:*

*cont\_search:*

*cmp si,di ;проверка на окончание (неуспешное):si>di*

*ja exit\_bad*

*mov bx,si*

*add bx,di*

*shr bx,1 ;делим на 2*

*cmp mas[bx],al ;сравниваем с искомым значением*

*je exit\_good ;искомое значение найдено*

*ja @@m1 ;mas[bx]>k*

*;mas[bx]<k:*

*mov si,bx*

*inc si*

*jmp cont\_search*

*@@m1: mov di,bx*

*dec di*

*jmp cont\_search*

*exit\_bad: mov result,0*

*jmp exit*

*nop ;вывод сообщения об ошибке*

*;выход*

*exit\_good:*

*mov result,bl*

*inc result ;вывод номера позиции найденного элемента*

*exit: mov ax,4c00h*

*int 21h*

*end main*

Строки

*n\_mas=$-mas*

*m\_mas db n\_mas dup (0)*

определяют длину исходного массива mas и присваивают ее значение переменной n\_mas, после чего резервируют область память для массива m\_mas размером, что и mas, заполненную байтами 0.

**ПРАКТИЧЕСКОЕ ЗАДАНИЕ**

Выполните а) линейный поиск в массиве целых чисел, б) поиск элемента в упорядоченной последовательности.

***Указание для случая б):*** когда выбираете диапазон значений для поиска, выражайте диапазон значениями индекса. Необходимо начинать поиск с установки трех переменных: первое, последнее, среднее, которые указывают на первое, последнее и среднее значения списка, соответственно.)

**Порядок выполнения работы.**

1. Загрузить программу эмулятор EMU8086
2. Набрать текст ***Программы 1***
3. Выполнить программу в пошаговом режиме.
4. Получить у преподавателя исходную последовательность и байт для поиска, выполнить с помощью программы поиск и предъявить результат преподавателю.
5. Набрать текст ***Программы 2*** и повторить для нее предыдущие действия.
6. Составить и зарисовать в отчете блок-схемы алгоритмов исследуемых программ.
7. Оценить скорость поиска данных по исследуемым алгоритмам. Для выполнения оценки подсчитать количество команд от начала выполнения программы до ее завершения (нахождения требуемого результата). Объяснить полученные результаты.
8. Сравнить результаты работы программы сортировки, написанной на базовом алгоритмическом языке, и на языке ассемблера. Сравнить быстродействие программ и объем загрузочного модуля.

**Содержание отчета.**

1. Наименование, цель и программу работы;
2. блок-схемы алгоритмов поиска, рассмотренных в лабораторной работе;
3. тексты и результаты работы программ поиска, сложения двух матриц и транспонирования прямоугольной матрицы;
4. выводы по работе.

**Контрольные вопросы**

1. Дать понятие массива данных.
2. Что понимают под сортировкой массива?
3. Назовите известные Вам простые методы сортировки.
4. Назовите улучшенные методы сортировки.
5. В чем суть "пузырьковой" сортировки?
6. Как реализуется алгоритм сортировки вставками?
7. В чем суть сортировки прямым выбором?
8. Объяснить принцип линейного поиска.
9. На чем строится механизм бинарного (двоичного) поиска элементов в массиве?
10. Какой из исследуемых алгоритмов поиска дает наибольшую скорость в работе?