Практика 6. Одномерные массивы

Недели семестра – 11, 12

6.3 Типовые задачи по обработке одномерных массивов (продолжение)

6.3 Типовые задачи по обработке одномерных массивов (продолжение)

Пример 10. Сортировка (упорядочивание) одномерного массива по возрастанию или по убыванию методом «пузырька».

Задача: дан одномерный массив, упорядочить массив по возрастанию элементов. То есть дан массив A=(a[0],a[1],...a[n-1]); требуется переставить его элементы так, чтобы a[0]<=a[1]<=a[2]<=...<= a[n-1].

Для упорядочения массива будем использовать метод "пузырька". Суть метода: просматриваем массив и переставляем элементы так, чтобы максимальный элемент выталкивался на правую границу массива (всплывал, как пузырек, на границе массива).

Вначале рассмотрим суть метода на примере следующего массива: n = 5, A=(32861).

1 просмотр - просматриваем последовательность 32861, сравнивая слева направо два соседних элемента и переставляя их, если надо:

1 сравнение: 32861 ---> 23861

2 сравнение: 23861

3 сравнение: 23861 ---> 23681

4 сравнение: 23681 ---> 23618

Наибольший элемент 8 "всплыл" в конце последовательности.

2 просмотр - просматриваем последовательность 2361, сравнивая слева направо два соседних элемента и переставляя их, если надо:

1 сравнение: 2361

2 сравнение: 2361

3 сравнение: 2361 ---> 2316

Наибольший элемент 6 "всплыл" в конце последовательности.

3 просмотр - просматриваем последовательность 231, сравнивая слева направо два соседних элемента и переставляя их, если надо:

1 сравнение: 231

2 сравнение: 231 ---> 213

Наибольший элемент 3 "всплыл" в конце последовательности.

4 просмотр - просматриваем последовательность 21, сравнивая слева

направо два соседних элемента и переставляя их, если надо:

1 сравнение: 21 ---> 12

Наибольший элемент 2 "всплыл" в конце последовательности.

Итак, для последовательности из n элементов число просмотров равно (n-1). В каждом к-ом просмотре производится (n-к) сравнений первых элементов. В результате каждого просмотра в конце текущей просматриваемой последовательности "всплывает" наибольший элемент.

Ниже дано подробное словесное описание полученного алгоритма.

1 просмотр. Последовательность из n элементов просматривается слева направо. Каждые 2 соседних элемента a[i] и a[i+1] сравниваются (то есть 0-ый и 1-ой, 1-ой и 2-ий и т.д.) и, если a[i] > a[i+1], то эти элементы переставляются. В результате наибольший элемент "всплывает" в конце рассматриваемой последовательности из n элементов.

2 просмотр. Последовательность из (n-1) первых элементов просматривается слева направо. Каждые 2 соседних элемента a[i] и a[i+1] сравниваются (то есть 0-ый и 1-ой, 1-ой и 2-ий и т.д.) и, если a[i] > a[i+1], то эти элементы переставляются. В результате наибольший элемент "всплывает" в конце рассматриваемой последовательности из (n-1) элементов.

....

(n-1)ый просмотр. Последовательность из n-(n-2)=2, то есть из 2-х элементов просматривается слева направо, то есть элементы a[0] и a[1] сравниваются и, если надо, переставляютя.

Ниже приведена программа, реализующая алгоритм упорядочивания массива по возрастанию методом пузырька (алгоритм упорядочивания по убыванию точно такой же, только в операторе сравнения if знак «>» заменяется на знак «<»). Обратите внимание на то, что в программе приведено два варианта записи циклов for, реализующих упорядочивания. Они выполняют одинаковую работу, хотя записаны немного по-разному. Этот факт показывает, что программы не следует заучивать наизусть, не понимая смысла: ведь одни и те же действия можно иногда записать по-разному. И еще одно замечание касается тестирования программ упорядочивания: их работу необходимо проверять на исходном массиве не менее, чем из трех элементов, причем элементы должны располагаться в порядке, противоположном тому, какой Вы хотите получить (например, для проверки упорядочивания по возрастанию надо взять исходный массив минимум из трех строго убывающих элементов, например, массив (3,2,1)).

```
/* Упорядочивание (сортировка) массива  по возрастанию методом пузырька */
#include <stdio.h>
int main()
{int  a[100],  /* Массив    */
      n,      /* Количество элементов в массиве */
      i,      /* i - счетчик сравнений в каждом просмотре, индекс сравниваемых элементов*/
      k,      /* к - номер просмотра                                    */
      c;      /* Вспомогательная переменная для перестановки 2-х элементов */
/* Ввод массива */
     do{
         printf("Введите количество элементов массива n, n<=100: ");
         scanf("%d",&n);
     }while(n<1||n>20);
     printf("Введите %d целых элементов: \n",n);
      for(i=0;i<n;i++) scanf("%d",&a[i]);
/* Сортировка массива  - 1-ый вариант записи циклов */
     for(k=1;k<n;k++)
       for(i=0;i<n-k;i++)
         if (a[i] > a[i+1])
           {c = a[i];
            a[i] = a[i+1];
            a[i+1] =c;
           }
/* Сортировка массива  - 2-ый вариант записи циклов */
     for(k=0;k<n-1;k++)
       for(i=0;i<n-k-1;i++)
         if (a[i] > a[i+1])
           {c = a[i];
```

```
            a[i] = a[i+1];
            a[i+1] =c;
          }
/*  Вывод массива после сортировки */
      printf("Упорядоченный массив: \n ");
         for(i=0;i<n;i++) printf("%d ",a[i]);
}
```

Пример 11. Общая постановка задачи. Формирование нового массива из некоторых элементов исходного массива.

Конкретная задача. Переписать элементы заданного целочисленного массива m, кратные 5, подряд в другой массив m5; если таких элементов в исходном массиве нет, выдать соответствующее сообщение.

Особенность этой задачи в том, что при формировании нового массива нельзя использовать тот же индекс, по которому просматриваем исходный массив. Иначе полученный массив будет «дырявым», часть элементов у него будет иметь неопределенные значения.

Пример. Пусть исходный массив имеет вид: m=(10, 1, 3, 5, 20). Тогда вид полученного массива будет: m5=(10, 5, 20). Обратите внимание на то, что один и тот же элемент со значением 5 в исходном массиве имеет индекс 3, а в новом массиве - индекс 1; элемент со значением 20 в исходном массиве имеет индекс 4, а в новом массиве - индекс 2.

Ниже приведена программа решения данной задачи.

```
/* Формирование нового массива m5 из элементов исходного массива m, кратных 5 */
#include <stdio.h>
int main()
{ int m[20],m5[20],   /* m – исходный массив, m5 – полученный массив */
      i,              /* i - индекс элементов в массиве  m  */
      n,              /* n – количество элементов в массиве m */
      k;              /* k : в процессе заполнения массива m5 – индекс следующего элемента, который будет записан в массив m5, по окончанию заполнения массива m5 - количество элементов в массиве m5 */
```

```
/* Ввод значений элементов массива m */
      do{
          printf(“Введите количество элементов массива n, n<=20: “);
          scanf(“%d”,&n);
      }while(n<1||n>20);
      printf(“Введите %d целых элементов: \n“,n);
       for(i=0;i<n;i++) scanf(“%d”,&m[i]);
/* Формирование массива m5 */
       k=0;
       for(i=0;i<n;i++)
          if (m[i]%5==0)  {m5[k]=m[i]; k++; }
/* Вывод: массива m5  или сообщения об отсутствии элементов, кратных 5, в массиве m */
       if (k==0)  puts(“В исходном массиве нет элементов, кратных 5”);
       else {puts(“Элементы, кратные 5:”);
              for(i=0;i<k;i++)
                  printf(“%d ”,m5[i]);
             }
}
```