Exercise 5: Arrays

PRAVEEN KUMAR R

Assignment	5
Reg No	312217104114
Name	PRAVEEN KUMAR R
Grade	
Date	27-02-2018

1 Array visitor

Polynomial evaluation:

polynomial $a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0$ is represented as an array. `a[0:n]` of its coefficients. Write a program to compute the value of a polynomial using Horner's rule. The crux of the algorithm is: (*accumulator*)

```
s = 0
for i = n-1 down to 0:
  s = s * x + a[i]
```

```
#include<stdio.h>
int main()
{
  float a[100],x;
  int n;
  scanf("%d",&n);
  for(int i=0;i<n;i++)
    scanf("%f",&a[i]);
  scanf("%f",&x);
  float s=0;
  for(int i=n-1;i>=0;i--)
    s=s*x+a[i];
  printf("%f",s);
}
```

```
547.0
```

Compare the algorithm with the algorithm for summing the items of an array.

Binary search:

We are given a sorted array of numbers. Define a function `binary_search()` to search for a number in the given sorted array.

Alogorithm

```
def binary_search(a,n,t):
   l=0
   h=n
   m=(l+n)//2;
   while(l!=n):
       if(a[m]==t):
           return m;
       elsif(a[m]>t):
           l=m+1
       else
           h=m
       m=(l+h)//2
   return n
```

Specification

```
binary_search(a, n, target)
```

that searches for `target` in `a[0:n]` using binary search algorithm. Let the function return an index `i` such that (*search*)

```
a[0:i] < target <= a[i:n]
```

```
#include<stdio.h>
int binary_search(int a[],int n,int t)
{
  int beg=0,end=n-1;
  int mid=(beg+end)/2;
  while(beg<=end)
    {
      if(a[mid]==t)
return mid;
      else if(a[mid]<t)
beg=mid+1;
      else
end=mid;
      mid=(beg+end)/2;
    }
  return -1;
}
int main()
{
  int a[100],n,t;
  scanf("%d",&n);
  for(int i=0;i<n;i++)
    scanf("%d",&a[i]);
  scanf("%d",&t);
  int pos= binary_search(a,n,t);
  printf("%d",pos);
}
```

```
5
```

2 Sorting

Selection sort

problem description

Selection sort: Selection sort is an algorithm for sorting an array of items, say `a[0:n]`. The idea of the algorithm is expressed below:

```
swap a[0], a[minimum(a,0,n)]
swap a[1], a[minimum(a,1,n)]
swap a[2], a[minimum(a,2,n)]
...
swap a[n-2], a[minimum(a,n-2,n)]
```

which uses `minimum(a, i, n)` to find the minimum of a subarray `a[i:n]`.

```
selection_sort (a, 0, n):
   for i = 0 to n-2:
      swap a[i], a[minimum(a, i, n)]
```

Implement selection sort, using `minimum()` function. Note: remember that when a function changes the items of an array parameter, the changes are effected in the items of the actual array argument also.

Test the function from `main()` for several lists of numbers. Each test should read a list of numbers from stdin.

Program

```
#include<stdio.h>
int minimum(int a[], int low, int high)
{
  int m=low;
  for(int i=low+1;i<high;i++)
    if(a[i]<a[m])
      m=i;
  return m;
}
void selectionsort(int a[],int n)
{
  int temp,min;
  for(int i=0;i<n-1;i++)
    {
      min=minimum(a,i,n);
      temp=a[min];
      a[min]=a[i];
      a[i]=temp;
    }
}

int main()
{
  int a[100],n;
  for( n=0; scanf("%d",&a[n])!=EOF;n++);
  for(int i=0;i<n;i++)
    printf("%d%c",a[i],i<n-1?',':'\n');
  selectionsort(a,n);
  for(int i=0;i<n;i++)
    printf("%d%c",a[i],i<n-1?',':'\n');
  return 0;
}
```

Test

- Input 24 -990 378 378 63 1 43 -98 382 3846 26 -727 173 2847
- Output

```
  24  -990  378  378  63   1  43  -98  382  3846   26  -727   173  2847
-990  -727  -98    1  24  26  43   63  173   378  378   382  2847  3846
```

3 Polish National Flag (PNF)*:

In an array of items `a[low:high]`, each item is either positive or negative. Define a function `partition(a, low, high)` that partitions the array into two subarrays `a[low:i]` and `a[i:high]` such that all the negative items of the array form `[low:i]`, and all the positive items form `[i:high]`. Test the function from `main()`. Use several lists of numbers for testing. (Note: We will use this algorithm for implementing `quicksort()`.)

Specification:

The partition algorithm takes an array `a[low:high]` as the input and returns an index `i` as the output such that all the negative items of the array form `[low:i]`, and all the positive items form `[i:high]`.

Algorithm development

The iterative step: After a few iterations, there are 3 subarrays, `[low:i]`, `[i:j]`, and `[j:n]`.

Initially, . . .

The next item to be scanned is `[j]`. There are two cases: `[j] < 0` and `0 <` [j]=.

Algorithm

```
partition a, low, high:
   i, j = low, high
   while j != high:
      if a[j] < 0:
         swap a[i], a[j]
         i = i+1
      j = j + 1
```

Program

```
#include <stdio.h>

int read_array (int a[]);
void print_array (int a[], int low, int high);
int partition (int a[], int n);
void swap (int a[], int i, int j);
```

```
int main ()
{
  int a[100];
  int n;
  int i;

  n = read_array(a);
  print_array (a, 0, n);
  i = partition (a, n);
  print_array (a, 0, i);
  print_array (a, i, n);
}


int read_array (int a[])
{
  int i;
  for (i = 0; scanf ("%d", &a[i]) != EOF; i++)
    ;
  return i;
}

void print_array (int a[], int low, int high)
{
  int i;

  for (i = low; i < high; i++)
    printf ("%d,", a[i]);
  printf ("\n");
}

int partition (int a[], int n)
{
  int i, j;

  i = 0;
  j = 0;
  while (j != n) {
    if (a[j] < 0) {
      swap (a, i, j);
      i++;
    }
    j++;
  }
  return i;
}
```

```
void swap (int a[], int i, int j)
{
  int t = a[i];
  a[i] = a[j];
  a[j] = t;
}
```

Test

Input

20 30 -80 -30 0 10 -40 -90 0 50 60 -50

Output

```
 20   30  -80  -30    0  10  -40  -90  0  50  60  -50
-80  -30  -40  -90  -50
 10   20   30    0   50  60    0
```

4 Dutch National Flag (DNF):

DNF is similar to PNF, but partitions the array `a[l:h]` into three subarrays `[l:i]`, `[i:j]` and `[j:h]`. Each item of the array has one of the three properties. Items having the same property should form one subarray each.

Specification

2 functions `print(a[l:h])`, used to print the array, `dnf()` which takes array `a[l:h]` and `c` as inputs and arrange the array based on `c`.

Prototype

```
void print(char a[], int l, int h);
int dnf(char a[], int l, int h, char c);
```

Program Design

The program contains 2 functions `print(char a[], int l, int h)`, which prints the array, `dnf(char a[], int l, int h, char c)`, which returns the index upto which the array has been rearranged, and `main()` which gets input from `stdin` and calls the functions.

Algorithm

```
def print(a[],l,h):
   for i in range(l,h):
      print(a[i])
```

```
def dnf(a[],l,h,c):
   i,p=l,l
   while i<h:
      if a[i]==c:
         a[i],a[p]=a[p],a[i]
         p+=1
      i+=1
```

Program

```
#include <stdio.h>

int read_array (int a[]);
void print_array (int a[], int low, int high);
int partition (int a[],int low, int n,int k);
void swap (int a[], int i, int j);

int main ()
{
  int a[100];
  int n;
  int i,j;

  n = read_array(a);
  print_array (a, 0, n);
  i = partition (a,0, n,-1);
  j = partition (a,i, n,0);
  print_array (a, 0, i);
  print_array (a,i,j);
  print_array (a, j, n);
}


int read_array (int a[])
{
  int i;
  for (i = 0; scanf ("%d", &a[i]) != EOF; i++)
    ;
  return i;
}

void print_array (int a[], int low, int high)
{
  int i;

  for (i = low; i < high; i++)
```

```
    printf ("%d,", a[i]);
  printf ("\n");
}

int partition (int a[], int low,int n,int k)
{
  int i, j;

  i = low;
  j = low;
if(k==-1)
 {
   while (j != n) {
    if (a[j] < 0) {
      swap (a, i, j);
      i++;
    }
    j++;
  }
}
else if(k==0)
{
while (j != n) {
    if (a[j] == 0) {
      swap (a, i, j);
      i++;
    }
    j++;
  }
}
  return i;
}

void swap (int a[], int i, int j)
{
  int t = a[i];
  a[i] = a[j];
  a[j] = t;
}
```

Test

Input

23 -90 67 -65 0 0 83 0 -282 56 -473 0 372 -34 56

Output

```
  23   -90    67   -65     0     0   83    0   -282   56   -473   0   372   -34   56
 -90   -65  -282  -473   -34
   0     0     0     0
  56    23    83   372    67    56
```