"Ss. Cyril and Methodius" University in Skopje
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Structured programming

Exercises 9

Version 1.0, 13 December, 2016

Table of Contents

1. Remainders from lectures

1.1. What is a pointer?

Pointer is a **data type** that stores (points to) some memory location. This memory location is stored by its address (some number).

1.2. How to declare pointer?

```
pointer_type *name;
```

Example

```
---
float *p;
---
```

⚠ This pointer doesn’t point to anything!

1.3. Why do we use pointers?

- For fast and efficient looping of complex data structures as arrays and trees.
- For efficient passing of complex arguments in functions. Passing array, struct or similar.
- For passing arguments whose values we want to stay affected after the function call.

2. Problem 1

Write a function that for an array of N integers will find the start and length of the largest increasing subarray.

Example

For array

2 3 **1 4 7 12** 7 9 1

will find the array with start index 2, and length of 4

Solution `p9_1_en.c`

```
#include <stdio.h>
#define MAX 100

void max_increasing(int x[], int n, int *pos, int *len) {
    int i, start, currLen;
    *pos = 0;
    *len = 1;
    for (i = 0; i < n - 1; i++) {
        start = i;
        currLen = 1;
        while ((x[i] < x[i + 1])) {
            currLen++;
            i++;
            if (i == n - 1) break;
        }
        if (currLen > *len) {
            *len = currLen;
            *pos = start;
        }
    }
}

int main() {
    int a[MAX];
    int i, n, pos, len;

    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

    max_increasing(a, n, &pos, &len);

    printf("Start: %d, Length: %d\n", pos, len);
    return 0;
}
```

3. Problem 2

Write a program that the input array \[a_0, a_1, … a_{n-1} \]

will transform into the output array: \[b_0, b_1, b_2, …, b_{n-1} \]

on the following way: \[b_0 = a_0 + a_{n-1} b_1 = a_1 + a_{n-2} . . . b_{n-1} = a_{n-1} + a_0 \]

Example

```
Input
1 2 3 5 7

should be transformed into
8 7 6 7 8
```

Solution `p9_2_en.c`

```c
#include <stdio.h>
#define MAX 100

void transform(int *a, int n) {
    int i, j;
    for (i = 0, j = n - 1; i < j; i++, j--) {
        *(a + i) += *(a + j);
        *(a + j) = *(a + i);
    }
    if (n % 2) {
        *(a + n / 2) *= 2;
    }
}
int main() {
    int i, n;
    int a[MAX];
    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for (i = 0; i < n; i++)
        printf("a[i] = %d\n", a[i]);
    transform(a, n);
    for (i = 0; i < n; i++)
        printf("b[i] = %d\n", a[i]);
    return 0;
}
```

4. Problem 3

Write the following functions for searching an array:

- Linear search
- Binary search

Then write a program that fill an array with numbers from 1 to 1000000, and than generates random number in this range and finds its index by calling the two functions for searching.

4.1. Homework

For the functions count and compare the number of needed iterations to find the number.

Solution `p9_3_en.c`

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 1000000

int linear_search(int *a, int n, int key) {
    int i;
    for (i = 0; i < n; i++) {
        if (*(a + i) == key) return i;
    }
    return -1;
}
int binary_search(int *a, int n, int key) {
    int start = 0;
    int end = n - 1;
    while (start <= end) {
        int mid = (start + end) / 2;
        if (*(a + mid) == key) return mid;
        else if (*(a + mid) > key) end = mid - 1;
        else start = mid + 1;
    }
    return -1;
}


int main() {
    int i;
    int *a = malloc(sizeof(int) * MAX);
    for (i = 0; i < MAX; i++) {
        *(a + i) = i + 1;
    }
    srand(time(NULL));
    int key = rand() % MAX + 1;
    printf("Element for search: %d\n", key);
    int found = linear_search(a, MAX, key);
    printf("Found with linear search at position: %d\n", found);
    found = binary_search(a, MAX, key);
    printf("Found with binary search at position: %d\n", found);
    return 0;
}
```

4.2. Homework

Write a recursive function for binary search.

5. Problem 4

Write functions for sorting array by using the following methods for sorting:

- Bubble sort
- Selection sort
- Insertion sort

Write functions fro reading and printing elements of an array, and write main program to test the sort functions.

5.1. Bubble sort

Solution `p9_4_en.c`

```c
void bubble_sort(int *a, int n) {
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (a[j] > a[j + 1])
                swap(&a[j], &a[j + 1]);
        }
    }
}
```

The method starts with comparing the first element with its neighbour until we reach the last element. In each comparison, if the previous has larger value, then they swap their values. That way the largest element is moved to the last position in the array. This method is repeated from 1-st to the last but one element in the array, so this position will float the element that is largest but smaller than the last element in the array. At the and we compare only the 1-st and 2-nd element of the array.

5.2. Selection sort

Solution `p9_4_en.c`

```c
void selection_sort(int a[], int n, int m) {
    if (n - m == 1)
        return;
    else {
        int smallest = a[m];
        int smallest_index = m;
        int i;
        for (i = m; i < n; ++i)
            if (a[i] < smallest) {
                smallest = a[i];
                smallest_index = i;
            }
        swap(&a[m], &a[smallest_index]);
        selection_sort(a, n, m + 1);
    }
}
```

We find the smallest element of the array and we swap the value of this element with the first element. After that, the first element is ignored (since its smallest) and recursively the method is repeated to rest of the array (second element, until the end). The recursion stops when there is only one element left. That is the final step, and the sorting is done.

5.3. Insertion sort

Solution `p9_4_en.c`

```c
void insertion_sort(int a[], int n) {
    int i, j;
    for (i = 1; i < n; i++) {
        int temp = a[i];
        j = i - 1;
        while (temp < a[j] && j >= 0) {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = temp;
    }
}
```

With this method we sort by inserting each element in appropriate position, hence the name of the sort. After the first iteration, the second element `a[1]` is compared to the first element `a[0]`. After the second iteration the third element is compared with first and the second. In general, in each iteration the element is compared with all elements in front of it. If the comparison shows that the element should be inserted at some position, then we make space by shifting all elements one place in right. This procedure is repeated for each element in the array.

5.4. Complete Solution

Solution p9_4_en.c

```
#include <stdio.h>

void bubble_sort(int *a, int n) {
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (a[j] > a[j + 1])
                swap(&a[j], &a[j + 1]);
        }
    }
}

void selection_sort(int a[], int n, int m) {
    if (n - m == 1)
        return;
    else {
        int smallest = a[m];
        int smallest_index = m;
        int i;
        for (i = m; i < n; ++i)
            if (a[i] < smallest) {
                smallest = a[i];
                smallest_index = i;
            }
        swap(&a[m], &a[smallest_index]);
        selection_sort(a, n, m + 1);
    }
}

void insertion_sort(int a[], int n) {
    int i, j;
    for (i = 1; i < n; i++) {
        int temp = a[i];
        j = i - 1;
        while (temp < a[j] && j >= 0) {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = temp;
    }
}

void insert(int a[], int n) {
    int i;
    for(i = 0; i < n; i++) {
        printf("a[%d] = ", i);
        scanf("%d", &a[i]);
    }
}
void print(int *a, int n) {
    int i;
    for(i = 0; i < n; i++) {
        printf("%d\t", *(a + i));
    }
    printf("\n");
}
int main() {
    int a[MAX], n;
    scanf("%d", &n);
    insert(a, n);
    bubble_sort(a, n);
    //selection_sort(a, n, 0);
    //insertion_sort(a, n);
    print(a, n);
    return 0;
}
```

6. Source code of the examples and problems

https://github.com/finki-mk/SP/

Source code ZIP