Аудиториски вежби 8
Покажувачи

М-р Ѓорѓи Маџаров
М-р Томче Делев

Структурирано програмирање

Факултет за информатички науки и компјутерско инженерство - Скопје 2011

Содржина

Покажувачи (Pointers)

Потсетување од предавања

Што е покажувач (pointer)?

Покажувач е податочен тип кој што чува (покажува кон) некоја мемориска локација. Оваа мемориска локација се чува преку нејзината адреса (некаков број).

Како се декларира покажувач?

```
pointer_type *name;
```

Пример:

```
float *p;
```

Вака декларираниот покажувач не покажува никаде!

Зошто служат покажувачите?

- За брзо и ефикасно изминување на сложени податочни структури како низи и дрва, ...
- За ефикасно пренесување на сложени аргументи во функции. Пренесување на низа, структура и слично
- За пренесување на аргументи чии што вредности сакаме да останат во онаа состојба во која се наоѓаат по извршувањето на функцијата

Задача 1

Да се напише функција која за низа од N цели броеви ќе ги пронајде почетокот и должината на најголемата растечка подниза.

Пример

За низата

2 3 1 4 7 12 7 9 1

ќе врати 2 4

Задача 1

Решение

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

void maxRastecka(int x[], int n, int *pos
     , int *len) {
    int i, start, currLen;
    start = 0;
    currLen = 1;
    *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) break;
        }
        if (currLen > *len) {
            *len = currLen;
            *pos = start;
        }
    }
}
```

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

    printf("Dolzina na nizata: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

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

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

Задача 2

Да се напише фунција која влезната низа $a_0, b_1, \ldots, b_{n-1}$ ќе ја трансформира во излезна низа $b_0, b_1, \ldots, b_{n-1}$ на следниот начин

$$
\begin{aligned}
b_0 &= a_0 + a_{n-1} \\
b_1 &= a_1 + a_{n-2} \\
&\vdots \\
b_n &= a_{n-1} + a_0
\end{aligned}
$$

Пример

Влезната низа

1 2 3 5 7

треба да се трансформира во

8 7 6 7 8

Задача 2

Решение

```c
#include "stdio.h"
#define MAX 100

void promena(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];
    printf("Kolku elementi ima nizata: ");
    scanf("%d", &n);
    printf("Vnesi gi elementite na nizata\n");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for (i = 0; i < n; i++)
        printf("a[i] = %d\n", a[i]);
    promena(a, n);
    for (i = 0; i < n; i++)
        printf("b[i] = %d\n", a[i]);
    return 0;
}
```

Задача 3

Да се напишат следните функции за пребарување во низа:

- Линеарно пребарување
- Бинарно пребарување

Потоа да се напише главна програма во која ќе се пополнува низа со броевите од 1 до 1 000 000, а потоа се генерира случаен број во овој опсег чија што позиција треба да се пронајде со повикување на двете функции за пребарување.
За дома: За двете функции избројте го и споредето го бројот на потребни итерации за проаноѓање на бројот.

Задача 3

Решение

```
int linearSearch(int *a, int n, int key) {
    int i;
    for(i = 0; i < n; i++) {
        if(*(a + i) == key) return i;
    }
    return -1;
}
int binarySearch(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;
}
```

За дома: Да се напише рекурзивна фунција за бинарно пребарување

Задача 3

Решение

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

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 shto se bara: %d\n", key);
    int found = linearSearch(a, MAX, key);
    printf("Najden so linearno preberavanje na pozicija: %d\n", found);
    found = binarySearch(a, MAX, key);
    printf("Najden so binarno preberavanje na pozicija: %d\n", found);
    return 0;
}
```

Задача 4

Да се напишат фунции за сортирање на низа со помош на следните методи за сортирање:

- Метод на меурче (Bubble sort)
- Метод со избор на елемент (Selection sort)
- Метод со вметнување (Insertion sort)

Да се напишат функции за внесување и печатење на елементите на една низа и да се напише главна програма во која се тестираат сите методи за сортирање.

Bubble sort

Задача 4

Се започнува од првиот елемент и се споредуваат секои два соседни додека не се дојде до последниот елемент. При секое споредување, ако претходниот има поголема вредност, тогаш си ги заменуваат местата. Така најголемиот елемент се доведува на последната позиција во низата. Се повторува истата постапка од 1-от до претпоследниот елемент во низата, така што сега на претпоследната позиција ќе исплива елемент помал од најголемиот елемент во низата итн. На крајот се споредуваат само 1-от и 2-от елемент од низата.

```
void bubbleSort(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]);
        }
    }
}
```

Selection sort

Задача 4

Се пронаоѓа најмалиот елемент во низата и истиот се заменува со првиот елемент. Потоа, првиот елемент на низата се игнорира (бидејќи се знае дека тој е најмал) и рекурзивно се сортира преостанатата подниза (од вториот елемент, па до крајот). Постапката се повторува сé дури не остане само еден елемент. Тоа е граничниот случај – се престанува со сортирањето

```
void selectionSort(int niza[], int n, int m) {
    if (n - m == 1)
        return;
    else {
        int najmal = niza[m];
        int indeksNajmal = m;
        int i;
        for (i = m; i < n; ++i)
            if (niza[i] < najmal) {
                najmal = niza[i];
                indeksNajmal = i;
            }
        swap(&niza[m], &niza[indeksNajmal]);
        selectionSort(niza, n, m + 1);
    }
}
```

Insertion sort

Задача 4

Со овој метод се сортира на тој начин што секој елемент се вметнува на соодветната позиција, од што и доаѓа самото име. Во првата итерација, вториот елемент a[1] се споредува со првиот елемент a[0]. Во втората итерација третиот елемент се споредува со првиот и вториот. Генерално, во секоја итерација елементот се споредува со сите елементи пред него. Ако при споредбата се покаже дека тој елемент треба да се вметне на соодветната позиција, тогаш се создава простор со поместување на сите елементи десно од тој елемент за еден и се вметнува елементот. Оваа процедура се повторува за секој елемент во низата.

```
void insertionSort(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;
    }
}
```

Задача 4

Решение

```
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;
    printf("Vnesi dolzina na nizata: ");
    scanf("%d", &n);
    insert(a, n);
    bubbleSort(a, n);
    //selectionSort(a, n, 0);
    //insertionSort(a, n);
    print(a, n);
    return 0;
}
```

Материјали и прашања

Предавања, аудиториски вежби, соопштенија
courses.finki.ukim.mk

Изворен код на сите примери и задачи
bitbucket.org/tdelev/finki-sp

Прашања и одговори
qa.finki.ukim.mk