Универзитет „Св. Кирил и Методиј“ - Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Структурно програмирање

Аудиториски вежби 9

Верзија 1.0, 27 Ноември, 2016

Содржина

1. Потсетување од предавања 1
    1.1. Што е покажувач 1
    1.2. Декларација на покажувач 1
    1.3. Пример 1
    1.4. За што служат покажувачите? 1
2. Задача 1 1
    2.1. Пример: 2
3. Задача 2 2
    3.1. Пример 3
4. Задача 3 3
    4.1. За дома: 4
    4.2. За дома: 4
5. Задача 4 4
    5.1. Bubble sort 5
    5.2. Selection sort 5
    5.3. Insertion sort 6
    5.4. Целокупното решение 6
6. Изворен код од примери и задачи 8

1. Потсетување од предавања

1.1. Што е покажувач

Покажувач е **податочен тип** кој што чува (покажува кон) одредена локација во меморијата. Оваа мемориска локација се чува преку нејзината адреса (позитивен цел број)

1.2. Декларација на покажувач

```
pointer_type *name;
```

1.3. Пример

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

Вака декларираниот покажувач не знаеме каде покажува!!!

1.4. За што служат покажувачите?

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

2. Задача 1

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

2.1. Пример:

За низата

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

ќе ја најде низата со почеток во индексот 2, со должина од 4

Решение `p9_1.c`

```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. Задача 2

Да се напише програма која влезната низа \[a_0, a_1, ... a_{n-1} \]

ќе ја трансформира во излезната низа: \[b_0, b_1, b_2, ..., b_{n-1} \]

на следниот начин: \[b_0 = a_0 + a_{n-1} b_1 = a_1 + a_{n-2} . . . b_{n-1} = a_{n-1} + a_0 \]

3.1. Пример

Влезната низа

```
1 2 3 5 7
```

треба да се трансформира во

```
8 7 6 7 8
```

Решение `p9_2.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. 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.

4.1. За дома:

За двете функции избројте го и споредете го бројот на потребни итерации за пронаоѓање на бројот.

Решение `p9_3.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. За дома:

Да се напише рекурзивна функција за бинарно пребарување на низа.

5. 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.

5.1. Bubble sort

Решение p9_4.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]);
        }
    }
}
```

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.

5.2. Selection sort

Решение p9_4.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);
    }
}
```

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

5.3. Insertion sort

Решение `p9_4.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;
    }
}
```

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.

5.4. Целокупното решение

Решение `p9_4.c`

```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. Изворен код од примери и задачи

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

Source code ZIP