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

Структурно програмирање

Аудиториски вежби 8

Верзија 1.0, 20 Ноември, 2016

Содржина

1. Рекурзивни функции

Функциите во програмскиот јазик Ц може да повикуваат други функции. Ова повикување може да оди во произволна длабочина.

Доколоку една функција се повикува сама себе, тогаш тоа се нарекува **рекурзивен повик**, односно тој начин на повикување се нарекува **рекурзија**

Пример за рекурзивна функција:

```
int factoriel(int n) {
    if (n==1) {
        return 1;
    }

    return n * factoriel(n-1);
}
```

1.1. Задача 1

Да се пресмета збирот: \[1! + (1+2)! + (1+2+3)! + … + (1+2+…+n)! \] притоа:

- користете **рекурзивна функција** за пресметување на збирот на првите k природни броеви.
- користете **рекурзивна функција** за пресметување на факториел на еден природен број.

Решение `p8_1.c`

```c
#include <stdio.h>

int factorial(int n) {
    if (n == 0)
        return 1;
    else
        return n * factorial(n - 1);
}

int sum(int k) {
    if (k == 0)
        return 0;
    else
        return k + sum(k - 1);
}

int main() {
    int i, n, result = 0;
    scanf("%d", &n);
    if (n > 0) {
        for (i = 1; i < n; i++) {
            int s = sum(i);
            result += factorial(s);
            printf("%d! + ", s);
        }
        int s = sum(n);
        result += factorial(s);
        printf("%d! = %d\n", s, result);
    } else
        printf("Wrong input\n");
    return 0;
}
```

1.2. Задача 2

Да се напише програма која за даден природен број ја пресметува разликата помеѓу најблискиот поголем од него прост број и самиот тој број.

Програмата треба да користи **рекурзивна** функција за наоѓање на соодветниот прост број, која пак треба да користи рекурзивна функција за проверка дали даден број е прост број.

1.2.1. Пример

Ако се внесе 573, програмата треба да испечати 577 - 573 = 4

Решение `p8_2.c`

```c
#include <stdio.h>

int is_prime(int n, int i);
int first_larger_prime(int n);

int main() {
    int number, difference;
    scanf("%d", &number);
    difference = first_larger_prime(number) - number;
    printf("%d - %d : %d\n", first_larger_prime(
                number), number, difference);
    return 0;
}
int is_prime(int n, int i) {
    if (n < 4)
        return 1;
    else if ((n % 2) == 0) return 0;
    else if (n % i == 0) return 0;
    else if (i * i > n) return 1;
    else return is_prime(n, i + 2);
}
int first_larger_prime(int n) {
    if (is_prime(n + 1, 3)) return n + 1;
    else return first_larger_prime(n + 1);
}
```

1.3. Задача 3

Да се напише програма што ќе ја испишува вредноста на n-тиот член на низата дефинирана со: \[\begin{array}{l} x_1 = 1\\ x_2 = 2\\ \vdots\\ x_n = \frac{n - 1}{n}x_{n - 1} + \frac{1}{n}x_{n - 2} \end{array} \]

Решение `p8_3.c`

```c
#include <stdio.h>
float xnn(int n) {
    if (n == 1)
        return 1;
    if (n == 2)
        return 2;
    return (n - 1) * xnn(n - 1) / n + xnn(n - 2) / n;
}

int main() {
    int n;
    scanf("%d", &n);
    printf("xnn(%d) = %.2f\n", n, xnn(n));
    return 0;
}
```

1.4. Задача 4

Да се напише рекурзивна функција која ќе го пресметува збирот на цифрите на еден број.

1.4.1. Пример:

```
sumDigits(126) -> 9
sumDigits(49) -> 13
sumDigits(12) -> 3
```

Решение `p8_4.c`

```c
#include <stdio.h>
int sum_digits(int n) {
    if (n == 0) return 0;
    return n % 10 + sum_digits(n / 10);
}
```

1.5. Задача 5

За даден број n, да се напише **рекурзивна функција** која ќе ги изброи појавувањата на цифрата 8. Притоа, доколку до некоја цифра 8 има уште една цифра 8 веднаш лево од неа, нејзиното појавување се брои двојно

1.5.1. Пример:

```
count8(8) -> 1
count8(818) -> 2
count8(8818) -> 4
```

Решение `p8_5.c`

```c
#include <stdio.h>

int count8(int n) {
    if (n == 0)
        return 0;
    if ((n / 10) % 10 == 8 && n % 10 == 8)
        return 2 + count8(n / 10);
    if (n % 10 == 8)
        return 1 + count8(n / 10);
    return count8(n / 10);
}
```

1.6. Задача 6

Да се напише програма која што за дадена низа од природни броеви (која што се внесува од тастатура) ќе го отпечати најголемиот заеднички делител (НЗД) на нејзините елементи. Програмата задолжително треба да содржи **рекурзивна функција** за пресметување на НЗД на два природни броја.

1.6.1. Пример

```
48 36 120 72 84
```

На екран треба да се отпечати:

```
NZD na elementite na ovaa niza e 12}
```

- НЗД за два броја може да се пресмета со користење на Евклидовиот алгоритам.
- За да се пресмета НЗД на броевите m и n, се пресметува остатокот при делење на m со n.
  - ако остатокот не е 0, се пресметува остатокот при делење на n со m % n
  - постапката се повторува се додека се добиваат ненеулти остатоци
  - ако остатокот е 0, НЗД на двата броја е последниот пресметан ненулти остаток.

1.6.2. Пример

```
NZD(20, 12)
20 % 12 = 8
12 % 8 = 4
8 % 4 = 0

NZD(20, 12) = 4
```

Решение `p8_6.c`

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

int NZD(int m, int n){
    if (!n) {
        return m;
    }

    return NZD(n, m % n);
}


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

    printf("Vnesi ja goleminata na nizata: ");
    scanf("%d", &n);

    printf("Vnesi gi elementite na nizata: \n");
    for (i=0; i<n; ++i){
        scanf("%d", &a[i]);
    }

    int nzd = NZD(a[0], a[1]);

    for (i=2; i<n; ++i){
        nzd = NZD(nzd, a[i]);
    }

    printf("NZD na elementite od nizata e %d",nzd);

    return 0;

}
```

1.7. Задача 7 (За Дома)

Да се напише програма која за дадена низа од цели броеви (која се внесува од тастатура), ќе го отпечати најмалиот заеднички содржател (НЗС) на нејзините елементи. Програмата треба задолжително да содржи рекурзивна функција за пресметување на НЗС на два броја.

1.7.1. Пример:

За низата: `18 12 24 36 6` на екран треба да се отпечати:

```
NZS na elementite na ovaa niza e 72
```

1.8. Задача 8

Да се напише програма која за дадена низа од цели броеви (која што се внесува од тастатура) ќе го отпечати збирот на елементите од низата. Програмата треба

да содржи рекурзивна функција за наоѓање на збирот на елементите во дадена низа.

Решение `p8_8.c`

```c
#include <stdio.h>

int sum_elements(int array[], int n);

int main() {
    int i, n, a[100];
    scanf("%d", &n);

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

    printf("The sum of all of the array elements is: %d \n", sum_elements(a, n - 1));

    return 0;
}

int sum_elements(int array[], int n) {
    if (n == 0) {
        return array[n];
    }
    else {
        return array[n] + sum_elements(array, n-1);
    }
}
```

1.9. Задача 9 (За Дома)

Да се напише програма која за дадена низа од цели броеви (која што се внесува од тастатура) ќе го отпечати најголемиот елемент. Програмата треба да содржи рекурзивна функција за наоѓање на најголем елемент во дадена низа.

1.9.1. Пример

За низата:

```
5 8 3 12 9 6
```

На екран треба да се отпечати:

```
Najgolem element vo nizata e 12
```

2. Изворен код од примери и задачи

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

Source code ZIP