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

Structured programming

Exercises 8

Version 1.0, 1 December, 2016

Table of Contents

1. Recursion

Functions in C can call other functions. This calling can go into arbitrary depth.

If one functions calls them self, then that call is called **recursive call** and the function is called **recursive** functions.

Example for recursive function:

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

1.1. Problem 1

Compute the sum: \[1! + (1+2)! + (1+2+3)! + ... + (1+2+...+n)! \] by:

- using **recursive function** for computing the sum of the first k natural numbers.
- using **recursive function** for computing the factorial of a natural number.

Solution `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. Problem 2

Write a program that for a given natural number will compute the difference between that number and the following prime number. The program should use **recursive** function for checking if a number is prime.

1.2.1. Example

For the number 573, the program should print 577 – 573 = 4

Solution p8_2_en.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. Problem 3

Write a function that will return the value of n-th member of the sequence defined with:

\[\begin{array}{l} x_1 = 1\\ x_2 = 2\\ \vdots\\ x_n = \frac{n - 1}{n}x_{n - 1} + \frac{1}{n}x_{n - 2} \end{array} \]

Solution p8_3_en.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. Problem 4

Write a recursive function that will compute the sum of the digits of a given number.

1.4.1. Example:

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

Solution `p8_4_en.c`

```c
#include <stdio.h>

int sum_digits(int n) {
    if (n == 0) return 0;
    return n % 10 + sum_digits(n / 10);
}
```

1.5. Problem 5

Given a non-negative int n, compute recursively (no loops) the count of the occurrences of 8 as a digit, except that an 8 with another 8 immediately to its left counts double, so 8818 yields 4.

1.5.1. Example:

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

Solution `p8_5_en.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. Problem 6

Write a program that for given array of integers (read from SI) will print the greatest common divisor (GCD) of its elements. GCD should be computed using recursive function.

1.6.1. Example

```
48 36 120 72 84
```

Should print:

```
GCD of elements of this array is 12
```

- GCD for two numbers can be computed using the Euclidean algorithm
- To compute GCD of numbers m and n, we compute the remainder of division of m with n
  - if remainder is not 0, we compute the remainder of division of n with (m % n)
  - this step is repeated until the remainder is zero
  - If the remainder is 0, GCD of the two numbers is the last non zero remainder.

1.6.2. Example

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

GCD(20, 12) = 4
```

Solution `p8_6_en.c`

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

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

    return GCD(n, m % n);
}


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

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

    int gcd = GCD(a[0], a[1]);

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

    printf("GCD of the elements of the array is %d",GCD);

    return 0;
}
```

1.7. Problem 7 (Try at home)

Write a program that for given array of natural numbers (read from SI) will find and print the least common denominator (LCD) of its elements. The program should use recursive function for computing LCD of two numbers.

Example:

For array: `18 12 24 36 6` the program should print:

```
LCD = 72
```

1.8. Problem 8

Write a program that fro given array of integers (read from SI) will print the smallest element. The program should use recursive function for finding the smallest element of an array.

Solution `p8_8_en.c`

```
Unresolved directive in sp_av8_en.adoc - include::../../src/av8/p8_8_en.c[]
```

1.9. Problem 9 (Try at home)

Write a program that for given array of natural numbers (read from SI) will compute the sum. The program should use recursive function for computing the sum of array of integers.

1.9.1. Example

For array:

```
5 8 3 11 9 6
```

The program should print:

```
sum is 42
```

2. Source code of the examples and problems

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

Source code ZIP