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

Lesson 7

Recursive functions

Structured Programming

Recursive functions

Remainders from lectures

- Functions in C can call other functions
  - This calling can go in some limited depth
- A function can call itself
- That way of calling function is called **recursion**

Example of recursive function

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

Problem 1

Compute the sum:
`1!+(1+2)!+(1+2+3)!+...+(1+2+...+n)!`
This time:

- Use **recursive** function to compute sum of the first k numbers
- Use **recursive** function to compute factorial of a number k

Problem 1
Solution

```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);
}
```

Problem 1
Solution

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

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 to check if number is prime.

Example

For the number 573, the program should print

```
577 - 573 = 4
```

Problem 2
Solution

```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);
}
```

Problem 3

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

$$
\begin{aligned}
x_1 &= 1 \\
x_2 &= 2 \\
&\vdots \\
x_n &= \tfrac{n-1}{n} x_{n-1} + \tfrac{1}{n} x_{n-2}
\end{aligned}
$$

Problem 3
Solution

```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;
}
```

Problem 4

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

Example

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

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

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.

Example

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

```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);
}
```

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.

Example

```
48 36 120 72 84
```

Should print:

```
GCD is 12
```

Euclidean algorithm

Problem 6

- 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.

Example

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

Problem 6
Solution

```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 is %d", gcd);
    return 0;
}
```

Problem 7

Try to solve 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 is 72
```

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.

Example

```
For array:
5 8 3 12 9 6
The result should be:
3
```

Problem 8
Solution

```c
#include <stdio.h>
int min_element(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("Smallest element is: %d \n", min_element(a, n - 1));
    return 0;
}
int min_element(int array[], int n) {
    if (n == 0)
        return array[n];
    else {
        int temp = min_element(array, n - 1);
        if (array[n] < temp)
            return array[n];
        return temp;
    }
}
```

Materials and Questions

Lectures, exsercises and announcements
courses.finki.ukim.mk

Source code of all examples and problems
https://github.com/tdelev/SP/tree/master/latex/src

Questions and discussion
forum.finki.ukim.mk