Data structures and algorithms– TP1

Prof. Maria Iuliana Dascalu
Assist. Iulia-Cristina Stănică
iulia.stanica@gmail.com

Grading

- The lab grade represents 50% of the final grade:
  - 30% – 3 big home assignments (you can work in teams of 2)
  - 20% – activity during labs (presence, activity, small homeworks)
- Small bonus for exceptional homeworks (> 50%)
- Attending at least 8 labs (requirement for taking the exam). Otherwise, you will have to redo the lab next year.

Grading and homeworks

- The labs and the 3 big assignments will be put on moodle.
- You will use the same platform to upload your big assignment solutions (respecting the deadline) and you **will have to present them** during the following lab. If you have problems with the moodle upload, you can also send the homework by email.
- Important! The big assignments will be checked against plagiarism!

- You can check your lab grades at the end of each week.

Questions

- For any question, you can contact me at the following email address:

iulia.stanica@gmail.com

- You can also use the forum: (http://fils.curs.pub.ro)

Tools

- C-Free 4.0 Standard (or 5.0 Professional) http://www.programarts.com/cfree_en/download.htm
- CodeBlocks (codeblocks-17.12mingw-setup.exe):

http://www.codeblocks.org/downloads/26

- Any other IDE or C / C++ compiler (e.g. GCC for Linux, Visual Studio)

Objectives

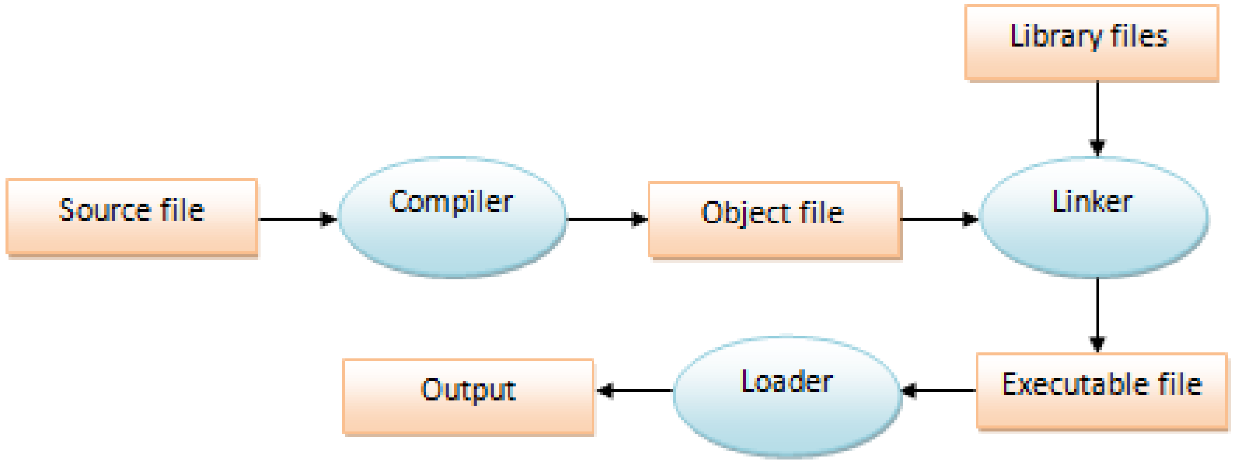
- to run and compile C programs;
- to identify the structure of a C program;
- to use standard I/O operations;
- to define variables;
- to declare and implement functions;

Typical C program

•A C program is written in a file with the ".c" extension: the source code.

•After compilation, another file, with the ".o" extension appears: the object code.

•After execution, another file, with the ".exe" extension appears: the executable.



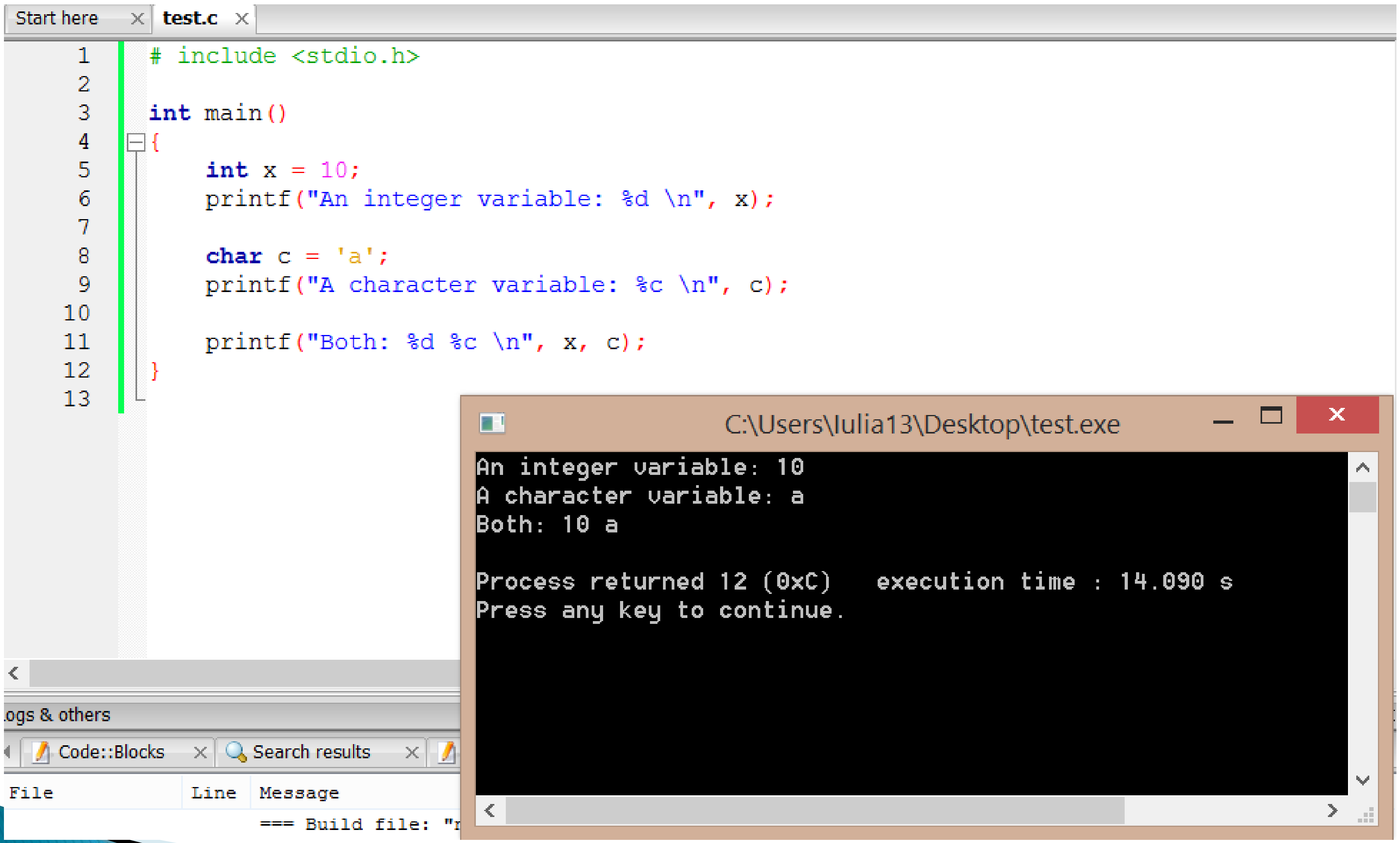
Typical C program

```
#include <stdio.h>
int main()
{
    printf( "Hello!\n" );
    getchar();
    return 0;
}
```

- Structure:
  - **Pre-processing directives** – tells the compiler to put code from the header into our program before actually creating the executable(eg: stdio.h – C library needed for using printf and scanf)
  - **One or more functions** (**main** is mandatory), with loops, if structure etc.

! Attention, C is case sensitive!

A. Standard Output Operations



Start here | test.c

```c
# include <stdio.h>

int main()
{
    int x = 10;
    printf("An integer variable: %d \n", x);

    char c = 'a';
    printf("A character variable: %c \n", c);

    printf("Both: %d %c \n", x, c);
}
```

C:\Users\Iulia13\Desktop\test.exe

```
An integer variable: 10
A character variable: a
Both: 10 a

Process returned 12 (0xC)   execution time : 14.090 s
Press any key to continue.
```

A. Standard Output Operations

- Format specifiers:

%i ou %d	int
%c	char
%f	float
%lf	double
%s	string

Signature of *printf* function

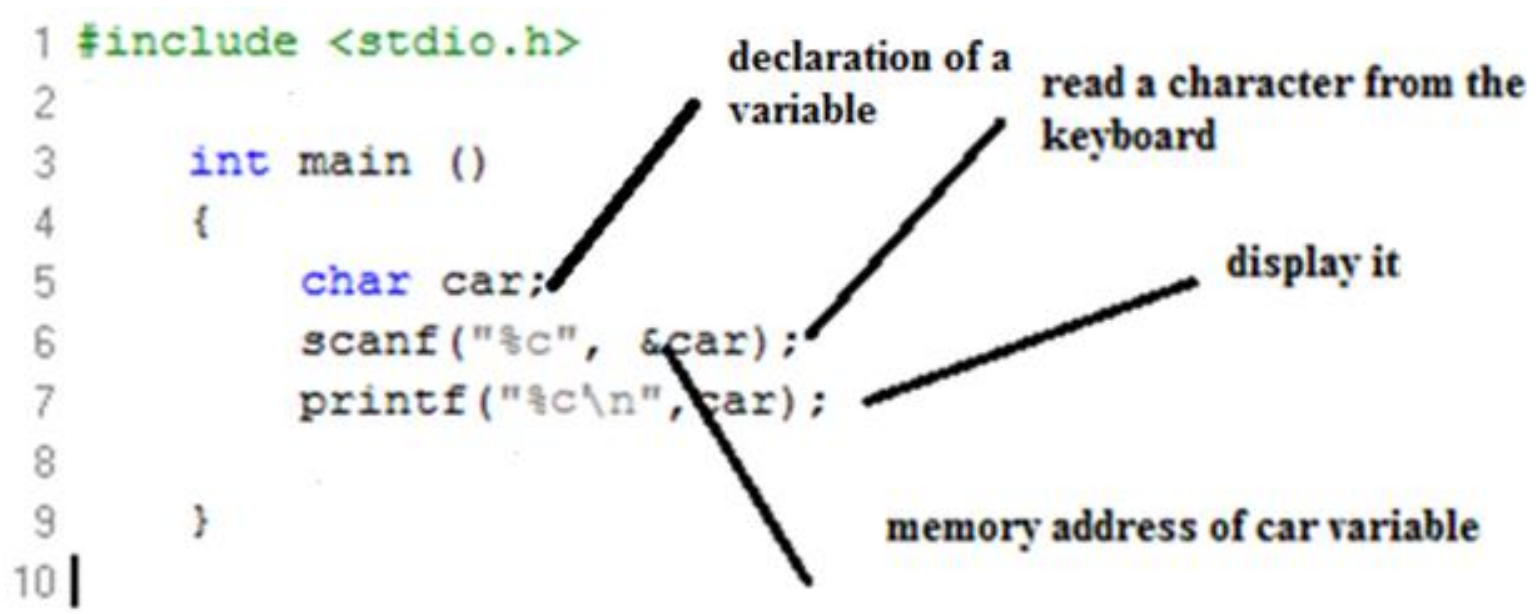
- **printf(control, par1, par2, …, parn);**

Where

- control = a string which defines the texts and the format parameters (between “ “)
- par1, par2, …, parn = expressions; their values are written taking into account the format parameters from control

<u>Exercice</u>: Test the format specifiers of the printf function

B. Standard Input Operations



```
#include <stdio.h>

    int main ()
    {
        char car;
        scanf("%c", &car);
        printf("%c\n", car);

    }

```

declaration of a variable
read a character from the keyboard
display it
memory address of car variable

- «Scanf» has the same signature as «printf» and is defined in «stdio.h» .

- Ex1. Write a program to calculate the average between two float numbers. The result shall be displayed with 2 decimals. Use *scanf* and *printf*!

- %.2f   -> format parameter for float with 2 decimals

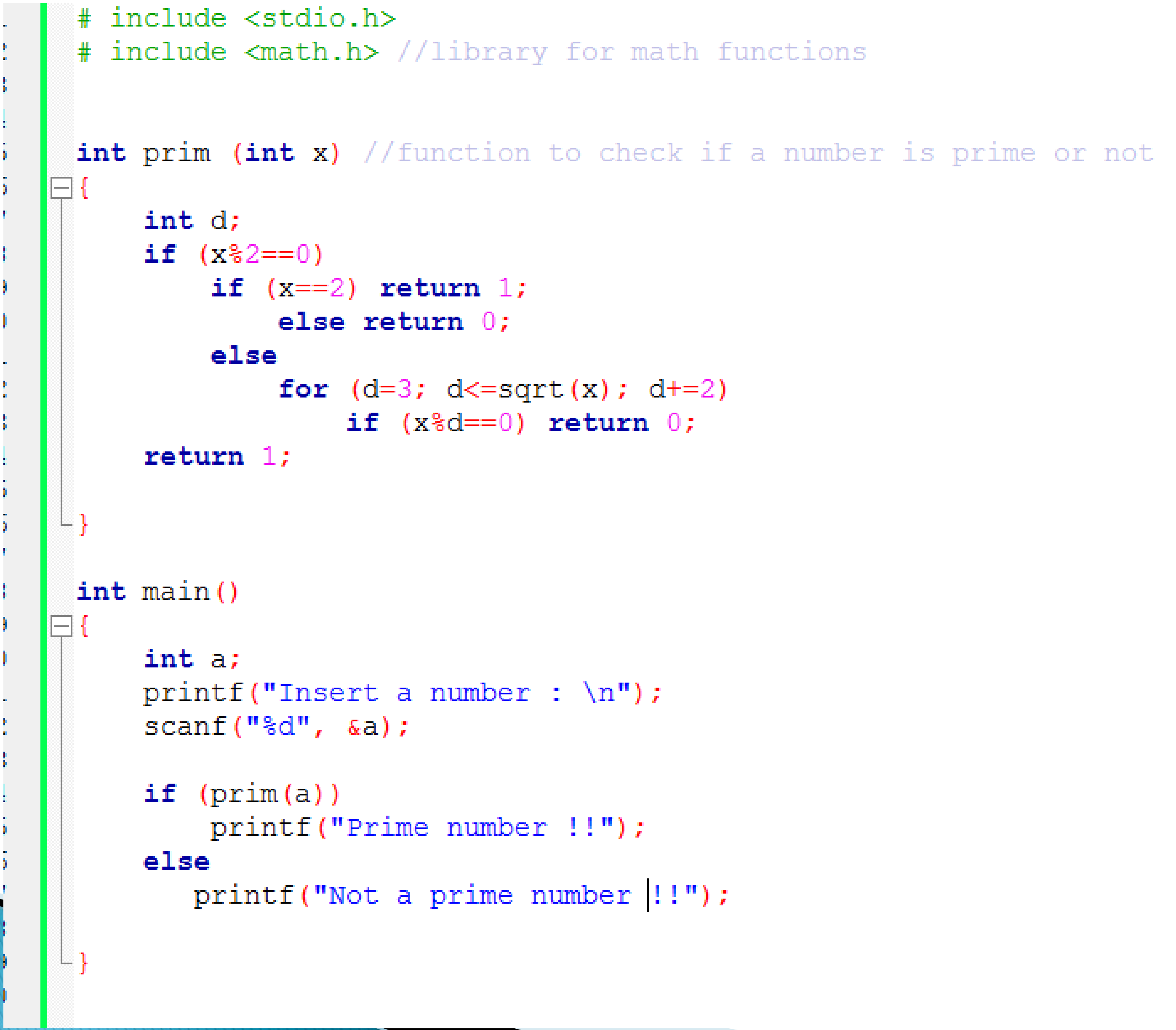
Functions: declaration and implementation

- Signature:

```
type_of_the_returned_result
  function_name(list_of_formal_params)
{
  declaration_of_local_variables;
  instructions;
}
```

- Visibility domain: local vs. global variables

Exemple functions



```c
# include <stdio.h>
# include <math.h> //library for math functions


int prim (int x) //function to check if a number is prime or not
{
    int d;
    if (x%2==0)
        if (x==2) return 1;
            else return 0;
        else
            for (d=3; d<=sqrt(x); d+=2)
                if (x%d==0) return 0;
    return 1;

}

int main()
{
    int a;
    printf("Insert a number : \n");
    scanf("%d", &a);

    if (prim(a))
        printf("Prime number !!");
    else
        printf("Not a prime number !!");

}
```

Exemple – observations

- Note the use of math.h library: for sqrt function (the same meaning as in Java)
- Note the control flow structures (if, if-else, for, …)
- Note the function definition and call: the implemented function calculates if a number is prime or not

Use functions for solving the following exercises:

- Ex2. Display the minimum of three float numbers, read with scanf.
- Ex3. Write a program which sums the digits of all numbers situated in a given interval. The endpoints of the interval are read from keyboard.
- Ex4. Write a program which reads an int number and checks if it’s a palindrome ot not (a palindrome number is symmetrical, ex: 131, 22122) .

Homework

- Ex1. Write a function primeNumbers which receives a number (n) as a parameter and displays the first n prime numbers. Test your function in the main.
- Ex2. Write a function « factorial » which receives a number (n) as a parameter and calculates its factorial. Test your function in the main.
- Ex3. Write a program which checks if two numbers are relatively prime. Use a function which receives the two numbers as parameters and test it in the main.

REFERENCES

- "C++ Programming Language", Bjarne Stroustroup
- "Thinking in C++", by Bruce Eckel & Chuck Allison
- "C++ Plus Data Structures", by Nell Dale
- "Limbajele C si C++ pentru incepatori"(vol 1-C, vol 2-C++), by Liviu Negrescu (romanian)
- Tutorials point: http://www.tutorialspoint.com/cprogramming/
- C Programming and C++ Programming: http://www.cprogramming.com/