



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

# Structured programming

Exercises 2

Version 1.0, 22 September, 2016

# Table of Contents

1. Simple C program structure.....	1
2. C Functions.....	1
3. Comments usage.....	1
3.1. Example (Hello world) .....	1
3.2. Program for summing two integers .....	2
4. Variables.....	2
4.1. Declaring variables .....	2
4.2. Data types in C .....	3
4.3. Defining variable names .....	3
5. Named constants .....	3
6. Operators .....	4
6.1. Arithmetic operators.....	4
6.2. Relational operators .....	4
6.3. Logical operators .....	4
6.4. Additional operators .....	5
6.5. Unary operators .....	5
7. Printing on the standard output.....	6
7.1. Most used format placeholders .....	6
7.2. Usage of function <code>printf</code> .....	6
8. Problems.....	7
8.1. Problem 1 .....	7
8.2. Problem 2.....	7
8.3. Problem 3.....	7
8.4. Problem 4.....	8
8.5. Problem 5.....	8
8.6. Problem 6.....	8
8.7. Problem 7.....	9
8.8. Problem 8.....	9
9. Source code of the examples and problems .....	11

# 1. Simple C program structure

All the source code written in C is organized in **functions**

```
int main() {  
    variable_declarations;  
    expressions;  
}
```

## 2. C Functions

- `main` - C **main** function
- In parentheses ( ) we put the input arguments
- The return type of the functions is before the name (`int` – the function returns integer)
- The function body starts with {, and ends with }
- All declarations and expressions are separated with ;

## 3. Comments usage

Comments are used for extra explanation or documenting the source code. C supports two types of comments:

- one line comments that start with double slash //

```
// comment in one line
```

- multiple line comments that can span across multiple lines and start with /\* and end with \*/

```
/*  
Longer comment  
in multiple  
lines  
*/
```

### 3.1. Example (Hello world)

## Example 1

```
#include <stdio.h>

// main function
int main() {
    /*
     * Printing a message on the standard output (the screen)
     */
    printf("Welcome to FINKI!\n");
    return 0;
}
```

- `#include` - directive for including external libraries
- `stdio.h` - library for accessing standard input/output streams (keyboard/screen)
- `printf` - function for printing on the standard output (screen)

## 3.2. Program for summing two integers

### Example 2

```
#include <stdio.h>

int main() {
    int a = 5;
    int b = 10;
    int c = a + b;
    return 0;
}
```

## 4. Variables

- Variables are symbolic names for places in memory where are stored some values.
- Before using it, each variable must be **declared**.
- With each assignment of new value, the old value in the variable is lost.

### 4.1. Declaring variables

```
data_type variable_name = initial_value;
```

#### Example

```
int a = 5;
float x = 2.3;
char c = 'a';
```

## 4.2. Data types in C

Integers	Characters	Real numbers
int	char	float
short		double
long		

## 4.3. Defining variable names

In naming variables you can use:

- lowercase letters from a to z;
- uppercase letters from A to Z;
- digits from 0 to 9 (the name can not start with digit);



When choosing a variable name, pick ones that clearly describe the value they store.



C is **case sensitive**

## 5. Named constants

Names constants are created using the keyword `const`

### *Example 3*

```
#include <stdio.h>

int main() {
    const long double PI = 3.141592653590L;
    const int DAYS_IN_WEEK = 7;
    const SUNDAY = 0; // by default int
    DAYS_IN_WEEK = 7; // error
    return 0;
}
```

Named constants can be created also by using the preprocessor and with all uppercase letters by convention.

Using `#define`

```
#define TEXT_TO_SEARCH_FOR REPLACEMENT_TEXT
```

*Example 4*

```
#include <stdio.h>
#define PI 3.141592653590L
#define DAYS_IN_WEEKS 7
#define SUNDAY 0

int main() {
    long number = PI;
    int day = SUNDAY;
    return 0;
}
```

## 6. Operators

### 6.1. Arithmetic operators

Are used on numbers (integers or real)

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo (residue after division)

### 6.2. Relational operators

Are used on all comparable types, and the result is an integer 0 (false) or 1 (true).

Operator	Meaning
<	Less then
<=	Less then or equal
>	Greater then
>=	Greater than or equal
==	Equality
!=	Non equality (different)

### 6.3. Logical operators

Used in combination with the relational operators to form complex logical expressions, that again return result 0 or 1.

Operator	Operation
&&	Logical AND
	Logical OR
!	Negation

## 6.4. Additional operators

Operator	Operation
=	Assignment
++	Increment (for +1)
--	Decrement (for -1)

## 6.5. Unary operators

The operators + and – can be used in unary manner:

*Changing the sign of the values stored in x and y*

```
x = +x;
y = -y;
```

### 6.5.1. Compound operators

Combination of assignment operator with other operator.

+=, -=, \*=, /=, %=

*Example of using compound operators*

```
a += 5; // a = a + 5;
a += b * c; // a = a + b * c;
```

*Example with variables and operators*

```
#include <stdio.h>

int main() {
    int a;
    float p;
    p = 1.0 / 2.0; /* p = 0.5 */
    a = 5 / 2; /* a = 2 */
    p = 1 / 2 + 1 / 8; /* p = 0.5 */
    p = 3.5 / 2.8; /* p = 1.25 */
    a = p; /* a = 1 */
    a = a + 1; /* a = 2 */
    return 0;
}
```

## 7. Printing on the standard output

For printing on the standard output (screen) in C we use the function `printf` from the library `stdio.h` (**S**tandard **I**ntput **O**utput)

```
#include <stdio.h>
```

The signature of the function is:

```
int printf(control_array, list_of_arguments)
```

The *control array* contains text of any kind, and format placeholders with leading `%` or special characters with leading `\`.

The *format placeholders* are determined from the variable type we want to print.

### 7.1. Most used format placeholders

Format	Usage
<code>%d</code>	integers (int)
<code>%i</code>	integers (int)
<code>%f</code>	real numbers (float, double)
<code>%c</code>	characters (char)
<code>%s</code>	characters (string, <code>char[]</code> , <code>char*</code> )
<code>%%</code>	character <code>%</code>

### 7.2. Usage of function `printf`

Print on the standard output the following sentences:

```
First sentence.
Second sentence.
Third sentence.
```

#### Example 5

```
#include <stdio.h>

int main() {
    printf("First sentence.\n");
    printf("Second sentence.\nThird sentence.\n");
    return 0;
}
```



*Пример 6*

```
#include <stdio.h>

int main() {
    printf(" is a word long %d letters.\n", printf("Macedonia"));
    return 0;
}
```

## 8. Problems

### 8.1. Problem 1

Write a program that will compute the value of the mathematical expression:  $x = 3/2 + (5 - 46 \cdot 5/12)$

*Solution 1*

```
#include <stdio.h>

int main() {
    float x = 3.0 / 2 + (5 - 46 * 5.0 / 12);
    printf("x = %.2f\n", x);
    return 0;
}
```

### 8.2. Problem 2

Write a program that for given value of  $x$  (during the declaration) will compute and print the value of  $x^2$ .

*Solution 2*

```
#include <stdio.h>

int main() {
    int x = 7;
    printf("Number %d squared is %d\n", x, x * x);
    return 0;
}
```

### 8.3. Problem 3

Write a program that for a given sides of one triangle, it will print the perimeter and area of the square (values are  $a = 5$ ,  $b = 7.5$ ,  $c = 10.2$ ).

*Solution 3*

```
#include <stdio.h>

int main() {
    float a = 5.0;
    float b = 7.5;
    float c = 10.2;
    float L = a + b + c;
    float s = L / 2;
    float P = s * (s - a) * (s - b) * (s - c);
    printf("Perimeter is: %.2f\n", L);
    printf("Area is: %.2f\n", P);
    return 0;
}
```

## 8.4. Problem 4

Write a program for computing the arithmetic mean of the numbers 3, 5 and 12..

*Solution 4*

```
#include <stdio.h>

int main() {
    int a = 3;
    int b = 5;
    int c = 12;
    float as = (a + b + c) / 3.0;
    printf("The arithmetic mean is %.2f\n", as);
    return 0;
}
```

## 8.5. Problem 5

Write a program that will print the remainder from the division of number 19 with 2, 3, 5 and 8.

*Solution 5*

```
#include <stdio.h>

int main() {
    int a = 19;
    printf("The residue of division with 2 is: %d\n", a % 2);
    printf("The residue of division with 3 is: %d\n", a % 3);
    printf("The residue of division with 5 is: %d\n", a % 5);
    printf("The residue of division with 8 is: %d\n", a % 8);
    return 0;
}
```

## 8.6. Problem 6

Write a program for computing and printing the circle area and perimeter. The circle

radius is read as decimal number.

### Solution 6

```
#include <stdio.h>
#define PI 3.14

int main() {
    float radius;
    scanf("%f", &radius);

    float perimeter = 2 * radius * PI ;
    float area = radius * radius * PI;
    printf("L = %f\n", perimeter);
    printf("P = %f\n", area);
    return 0;
}
```

## 8.7. Problem 7

Write a program that reads from standard input two integers and prints their sum, difference, product and division remainder.

### Solution 7

```
#include <stdio.h>

int main() {
    int x, y;
    scanf("%d %d", &x, &y);

    printf("%d + %d = %d\n", x, y, x + y);
    printf("%d - %d = %d\n", x, y, x - y);
    printf("%d * %d = %d\n", x, y, x * y);
    printf("%d %% %d = %d\n", x, y, x % y);
    return 0;
}
```

## 8.8. Problem 8

Write a program that reads uppercase letter from standard input and prints out in lowercase.



Each character is represented with its ASCII code.

### example

```
A = 65, a = 97
```

## *Solution 8*

```
#include <stdio.h>

int main() {
    char c;
    printf("Enter an uppercase letter: ");
    scanf("%c", &c);
    printf("%c lowercase is: '%c'\n", c, c + ('a' - 'A'));
    return 0;
}
```

## 9. Source code of the examples and problems

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

Source code ZIP