SEMINAR 1

Contents

1. OBJECTIVES

- get familiar with the C programming language
- solve simple problems
- procedural programming in C

2. BASIC C NOTIONS

THE FORM OF A C PROGRAM

- all C program must contain at least one function: main – the first function that is called when the program executes
- all C statements must end with a semicolon (;)

```
pre-processor directives

global declarations

T f1()
{
        local variables' declarations;
        C statements;
}
```

```
T f2()
{
        local variables' declarations;
        C statements;
}
……

int main()
{
        local variables' declarations;
        C statements;

        return 0;
}
```

Pre-processor directives:

- the preprocessor is a separate program invoked by the compiler as the first part of the translation
- always begin with the hash symbol "#" ()
- tell the preprocessor to perform specific actions:
    - replace tokens in the text (#define)
    - insert the contents of other files into the source file (#include)
    - control compilation of portions of a source file (#ifdef, #ifndef, #else, #endif)

DATA TYPES

- int, float, double, char, void

VARIABLE DECLARATIONS AND INITIALIZATION

- int a, b;
- float x = 4.5;
- char s = 'a';

INPUT AND OUTPUT FUNCTIONS

- scanf (stdio.h)
- printf (stdio.h)

3. SIMPLE PROGRAMS

1. **Write a program which prints the sum of two given integer numbers.**

```c
#include <stdio.h>

int main()
{
	int m = 0;
	int n = 0;
	int sum = 0;

	printf("Input the first number: ");
	scanf("%d", &m);

	printf("Input the second number: ");
	scanf("%d", &n);

	sum = m + n;
	printf("The total is: %d.\n", sum);

	return 0;
}
```

2. **Write a program which asks for your name and surname. The application will print a greeting containing your entire name, as well as the number of characters your entire name contains.**

```c
int main()
{
	char surname[50];

	printf("Enter your surname, in lowercase: ");
	scanf("%49s", surname);
	surname[0] = toupper(surname[0]);

	char firstName[100];
	printf("Enter your first name: ");
	scanf("%49s", firstName);
	firstName[0] = toupper(firstName[0]);

	strcat(firstName, " "); // add a space to the  first name
	printf("Hello, %s! :)\n", strcat(firstName, surname));
	printf("Your entire name contains %d characters.\n", strlen(firstName)); //
firstname now contains both names + the space

	return 0;
}
```

4. PROCEDURAL PROGRAMMING IN C - FUNCTIONS

1. **Create a C function that receives as input 2 integer numbers and returns their sum and their product.**

Observation: In C, to return more than one value from a function one should use pointers, passed as arguments to the function. Please see below.

```
/*
	Computes the sum and product of 2 given numbers.
	Input: a, b - integer numbers
	Output: s, p - integer numbers representing the sum and product of a and b.
*/
void sumAndProduct(int a, int b, int* s, int* p)
{
	*s = a + b;
	*p = a * b;
}

int main()
{
	int a = 0;
	int b = 0;

	printf("Enter a and b: ");
	scanf("%d %d", &a, &b);

	int s = 0;
	int p = 1;
	sumAndProduct(a, b, &s, &p);
	printf("The sum is: %d and the product is: %d.\n", s, p);

	system("pause");

	return 0;
}
```

2. **Write a C application with a menu based console interface which:**
   a. **Reads a sequence of integer numbers, until 0 in encountered and prints the sum of all read numbers.**
   b. **Given a vector of numbers, finds the longest contiguous subsequence such that all elements are equal.**

Each requirement must be resolved using at least one function. All functions need to be specified.