PROGRAMMING LANGUAGES

Laboratory 4

INSTRUCTIONS (part II)

I. THEORETICAL BACKGROUND

1. Instructions (overview)

1.1. The conditional instruction (if..else)

```
if(expresie)
      instruction1;
else
      instruction2;
```

Obs.: "else" is linked to the last encountered "if".

```
E.g.:   if (exp1) instr1;
        else   if (exp2)  instr2;
                  else   if (exp3) instr3;
                            else instr4;
```

1.2. The "while" instruction

while (expression) instruction;

The instruction is recurrently executed as long as the expression's value is not 0.
The test takes place at the beginning, therefore if at the first check the expression is 0, the instruction from the while body does not take place at all.

1.3. The "do...while" instruction

do instruction while (expression);

The instruction is recurrently executed until the expression becomes 0. The test takes place after each execution of instruction, therefore the instruction will take place at least once.

1.4. The "for" loop:

```
for (exp1; exp2; exp3)
    instruction;
```

where: exp1 => initialises the loop (optional)
exp2 => the end test of the loop (optional)
exp3 => updating the index variable (optional)

The for loop is equivalent to:

```
exp1;
while (exp2)
    {   instruction;
        exp3;

    }
```

1.5. The "break" instruction:

```
break;
```

This instruction forcedly terminates any "while", "do...while", "for" or "switch" instruction that contains it. The program then jumps to the following instruction. This instruction is not used within "if...else" instructions or directly inside a function.

1.6. The "switch" instruction (for commutation):

```
switch (exp)
    { case exp1 : instrution set 1
        break;
      case exp2 : instruction set 2
        break;
      default : instruction set
    }
```

2. The standard function "exit"

The function prototype: **void exit (int code);**

The function is located in the header files: stdlib.h and process.h
When calling this function, the following actions take place:

* all the opened files buffers are erased
* all opened files are closed;
* the execution of the program halts.

The parameter of this function defines the program state at the moment of call. The 0 value defines a normal termination of the program execution, and a non-zero value signals the presence of an error (abnormal program termination).

Therefore, the "exit" function could be called in order to terminate a program, no matter if the program is aborted normally or due to an error occurrence.

II. ASSIGNMENT WORKFLOW

1. Take the function

$$y = 3*x*x + 2*x - 10 \quad \text{if } x > 0$$

and

$$y = 5*x + 10 \quad \text{if } x <= 0$$

Write a program that reads the value of "x", and then calculates and displays the resulting "y".

A possible solution:

```
#include <stdio.h>
main()  /* reads x, calculates and displays the value of y defined as follows:
                        y = 3*x*x + 2*x - 10      if x > 0
                        y = 5*x + 10              if x <= 0   */
{
  float x, y;
  printf ("\nIntroduce x: ");
  scanf ("%f", &x);
  if (x>0)
    y = 3*x*x + 2*x - 10;
  else
    y = 5*x + 10;
  printf ("x = %f\ty = %f\n", x, y);
getch ();
}
```

2. Modify the program from point 1 so as for the "if...else" instruction to be replaced with the conditional operator.

3. Write a program that calculates and displays the values of the function:

$f(x) = 5*x*x + 3*x - 7$
for $x=1, 2, ..., 10$

A possible solution:

```
#include<stdio.h>

main()   /*displays the values of the function f(x)=5*x*x + 3*x – 7 for  x=1, 2, 3,... , 10  */
 {
  int x;
  x = 1;
  while (x <= 10)
      {
       printf("\nx=%d\tf(x)=%d\n", x, 5*x*x + 3*x - 7);

       x=x+1;  /* better write x++, it is shorter*/

      }  /*end while*/
  getch();
 }  /*end main*/
```

4. Modify the program from above so as for the "while" body to contain a single instruction.

5. Write a program that reads an integer n, then calculates and displays n!.

We have

$$n! = 1*2*...*(n-1)*n$$

This calculus implies a cyclic process such as:

```
f=1 and i=1
while (i<=n)
      {
       f=f*i;
       i=i++;
      }
```

The program also needs a sequence for avoiding undesired values, which will be treated using the function "exit".

A possible solution:

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

main() /* reads n from the interval [1, 170],
            determines and displays n! */

  {
    int n, i;
    double f;

    printf ("\nValue of n:");
    if (scanf ("%d",&n) != 1)  /* the decimal values are converted to %d */
       {
        printf ("\n No number has been introduced\n");
        getch( );
        exit(1);
       }
```

```
    if (n<0 || n>170)
       {
        printf("\n n doesn't belong to the interval [0, 170]\n");
        getch( );
        exit(1);
       }
    f=1.0;
    i=2;
    while (i<=n)
        f=f*i++;
        printf ("\n n=%d\t n!=%g\n", n, f);
    getch();
 }
```

6. Modify the program from point 5 so as to recurrently read the n values and determine n! values, without leaving the program. For this purpose, a "while" cycle shall be used.

7. Modify the program from point 6, so as to use "for" instead of "while" when computing the factorial value.

8. Modify the program from point 6, so as to recurrently read the n values and determine the resulting n! values, without leaving the program. For this purpose, a "do...while" cycle shall be used.

9. Write a program (with avoiding sequences similar to the one from point 5) that determines a to the power of n using a "for" loop.

10. Write a program that reads a digit from the interval [1,7] and displays the name of its corresponding week day (1-Monday, 2-Tuesday, etc.).

III. SOLUTIONS

2. L4_2.C

```
#include <stdio.h>

main()  /* reads x, determines and displays y defined as:
                                                    y = 3*x*x + 2*x - 10      if  x > 0
                                                    y = 5*x + 10              if  x <= 0   */

 {
  float x, y;
  printf("\nIntroduce x: ");
  scanf("%f", &x);
  y = x>0 ? 3*x*x + 2*x - 10 : 5*x + 10;
  printf ("x = %f\ty = %f\n", x, y);
  getch( );
 }
```

4. L4_4.C

```
#include<stdio.h>

main( )   /*displays the values for function f(x)=5*x*x +3*x -7
                                        for x=1, 2, 3,... , 10  */
 {
  int x;
  x = 0;
  while (++x <= 10)
      {
       printf ("\nx=%d\tf(x)=%d\n", x, 5*x*x + 3*x - 7);
      } /* end while */
  getch( );
 }  /*end  main*/
```

6. L4_6.C

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

main( )    /* reads n from interval [1, 170] and determines n! */
  {
    int n, i;
    char stop;
    double f;
    stop=1;
    while (stop!= 'y')
    {
    printf ("\nValue of  n: ");
    if(scanf("%d",&n) != 1)    /* decimal values are turned to %d when reading */
       {
        printf ("\n No number has been introduced\n");
        getch(                                                                                    );
        exit(1);
       }
```

```
    if (n<0  ||  n>170)
       {
        printf ("\n n does not appear in [0,170]\n");
        getch( );
        exit(1);
       }


    f=1.0;
    i=2;
    while (i<=n)
        f=f*i++;
        printf ("\n n=%d\t n!=%g\n", n, f);
    printf ("\n Do you wish to leave the program? (y/n)\n");
    stop=getch( );
   }
  }
```

7. L4_7.C

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

main( ) /* reads n from the interval [1, 170], calculates and displays n! */
  {
    int n, i;
    char stop;
    double f;
    stop=1;
    while (stop!='y')
    {
    printf ("\nValue of n: ");
    if (scanf("%d", &n) != 1)
       {
        printf("\n no number has been introduced\n");
        getch( );
        exit(1);
       }
    if(n<0  ||  n>170)
       {
        printf ("\n n doesn't belong to [0, 170]\n");
        getch( );
        exit(1);
       }
    for(f=1.0, i=2; i<=n; i++)
        f *= i;
        printf("\n n=%d\t n!=%g\n", n, f);
    printf("\nDo you wish to leave the program? (y/n)\n");
    stop=getch( );
   }
  }
```

8. L4_8.C

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

```
main( ) /* reads n from [1, 170], calculates and displays n! */
  {
   int n, i;
   char stop;
   double f;
   do
   {
   Printf ("\nValue of  n: ");
   if (scanf("%d", &n) != 1)
     {
      printf ("\n No number has been given.\n");
      getch( );
      exit(1);
     }
   If (n<0  ||  n>170)
     {
      Printf ("\n n doesn't appear in [0, 170]\n");
      getch( );
      exit(1);
     }
   For (f=1.0, i=2; i<=n; i++)
      f *= i;
      printf ("\n n=%d\t n!=%g\n", n, f);
   printf ("\nDo you wish to leave the program? (y/n)\n");
   stop=getch( );
  }while (stop!='y');
 }
```

9. L4_9.C

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

main( )  /* reads a and n, calculates and displays a**n  */
{
 int i, n;
 long double a, p;  /* long double has the format specifier %Lf */

 printf ("\nThe base a =  ");
 if (scanf ("%Lf",&a)!=1)
       {
        printf ("\nNo number has been given \n");
        getch( );
        exit(1);
       }
 printf (" The exponent n = ");
 if (scanf ("%d",&n) != 1 || n<0)
       {
        printf ("Nu positive integer has been given ");
        getch( );
        exit(1);
       }
 for (i=0, p=1.0; i<=n; i++)
       p *= a;
 printf("a=%Lf\tn=%d\ta**n=%Lf\n", a, n, p);
```

```
 getch( );
}
```

10. L4_10.C

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

main( )  /* reads a digit belonging to [1, 7] and displays the name of its corresponding week day */
{
 int i;
 puts ("Write a digit from the interval [1, 7].");
 scanf ("%d",&i);
 switch(i)
 {
  case 1: puts ("Monday");
        break;
  case 2: puts ("Tuesday");
        break;
  case 3: puts ("Wednesday");
        break;
  case 4: puts ("Thursday");
        break;
  case 5: puts ("Friday");
        break;
  case 6: puts ("Saturday");
        break;
  case 7: puts ("Sunday");
        break;
  default: puts ("Tasta gresita");
        break;
 }
 getch( );
}
```