1.6 - The Preprocessor

All exercises in this Level must be coded *exclusively* in C syntax (no <iostream>, cout, cin, classes, etc.)

Exercise 1

Write a C-program that contains two print macro calls. The first prints the variable *a*, the second prints the variables *a* and *b*. Printing happens by the use of the PRINT1 and PRINT2 macros that accept arguments. These macros must be defined in an include-file. The variables *a* and *b* gets their value in the function *main()*.

Name the program “Macro.c” and the include-file “Defs.h”. Don’t forget to implement the mechanism to avoid multiple inclusion of the header file.

Exercise 2

Create the two macros MAX2(x,y) and MAX3(x,y,z). These macros must return the maximum value of the given arguments. Let the macro MAX3 make use of the macro MAX2. Add these macros to the file “Defs.h”.

1.7 - Pointers and Arrays

Exercise 1

Try to create a function *Swap()*. This function must exchange the value of two variables. For example: if *i=123* and *j=456*, then *i=456* and *j=123* after the *Swap()* function has been called. The variables *i* and *j* are declared, initialised and printed in the function *main()*. This problem can be solved by using pointers as arguments for the *Swap()*function.

Exercise 2

The following program reads a string with a 30 character maximum. Implement the *Length()* function. The function *Length()* must determine the length of the string. Give *Length()* the address of the array as argument.
Note: your *Length()* function should be similar to the built-in *strlen()* function so your job is to mimic that function without using it.

EOF is used in the function *main()*. This means End-of-File and is discussed later on in this document.
In DOS, EOF can be entered by the key combination Ctrl-z (often written as ^Z). With ^Z (Say: control Z) is meant pressing the control-key and the z-key simultaneously.

```
/* Calculate the length of a string */
#include <stdio.h>
#define MAXLINE 30
// String lenght declaration
int Length(char str[]);

int main()
{
     char string[MAXLINE+1];  // Line of maxium 30 chars + \0
     int c;                   // The input character
     int i=0;                 // The counter

     // Print intro text
     printf("Type up to %d chars. Exit with ^Z\n", MAXLINE);

     // Get the characters
     while ((c=getchar())!=EOF && i<MAXLINE)
     {
          // Append entered character to string
          string[i++]=(char)c;
     }
     string[i]='\0';          // String must be closed with \0
     printf("String length is %d\n", Length(string));
     return 0;
}
/* Implement the Length() function here */
```

Exercise 3

```c
/* Predict what will be printed on the screen */

#include <stdio.h>

#define PRD(a) printf("%d", (a) )       // Print decimal
#define NL          printf("\n");       // Print new line

// Create and initialse array
int a[]={0, 1, 2, 3, 4};

int main()
{
     int i;
     int* p;

     for (i=0; i<=4; i++) PRD(a[i]);                    // 1
     NL;

     for (p=&a[0]; p<=&a[4]; p++) PRD(*p);              // 2
     NL;
     NL;

     for (p=&a[0], i=0; i<=4; i++) PRD(p[i]);           // 3
     NL;

     for (p=a, i=0; p+i<=a+4; p++, i++) PRD(*(p+i));    // 4
     NL;
     NL;

     for (p=a+4; p>=a; p--) PRD(*p);                    // 5
     NL;

     for (p=a+4, i=0; i<=4; i++) PRD(p[-i]);            // 6
     NL;

     for (p=a+4; p>=a; p--) PRD(a[p-a]);                // 7
     NL;

     return 0;
}
```

Exercise 4

Create a C-program that has a function DayName()which can print the day of a given day-number. For example:

1 gives: Day 1 is a Sunday
7 gives: Day 7 is a Saturday.

The day-name (1-7) should be written "hard-coded" into the program using an array of strings.

1.8 - Structures

Exercise 1

Write a C-program that prints the contents of a struct called *Article*. An *Article* has the following characteristics:

- Article number
- Quantity
- Description (20 characters)

The test program must create an *Article* of which the contents are assigned at initialization level.

Printing the *Article* is done with a *Print()* function. This function gets the address of the structure as a parameter.

Tip: Suppose that *p* is the pointer to the structure. It will allow the fields to be printed by *(*p).fieldname* or *p->fieldname*.

1.9 - Input and Output

Exercise 1

Create a C-program that reads the characters from the keyboard and shows them on screen (the inputted characters should only be displayed when the user hits 'enter', line by line).

When ^A is entered, the program must end properly. Then the following message will appear: “CTRL + A is a correct ending.”

Tip: *getchar()* reads and *putchar()* writes the type *int*. The value of ^A is 1.

Exercise 2

Alter the last program of exercise 1 in such a way that the output doesn’t go to the screen but is written to a file. The file to write to must be specified by the user.