université de science et technologie houari boumediene

TP 6 : Functions in C

beldjouziasmaa@gmail.com

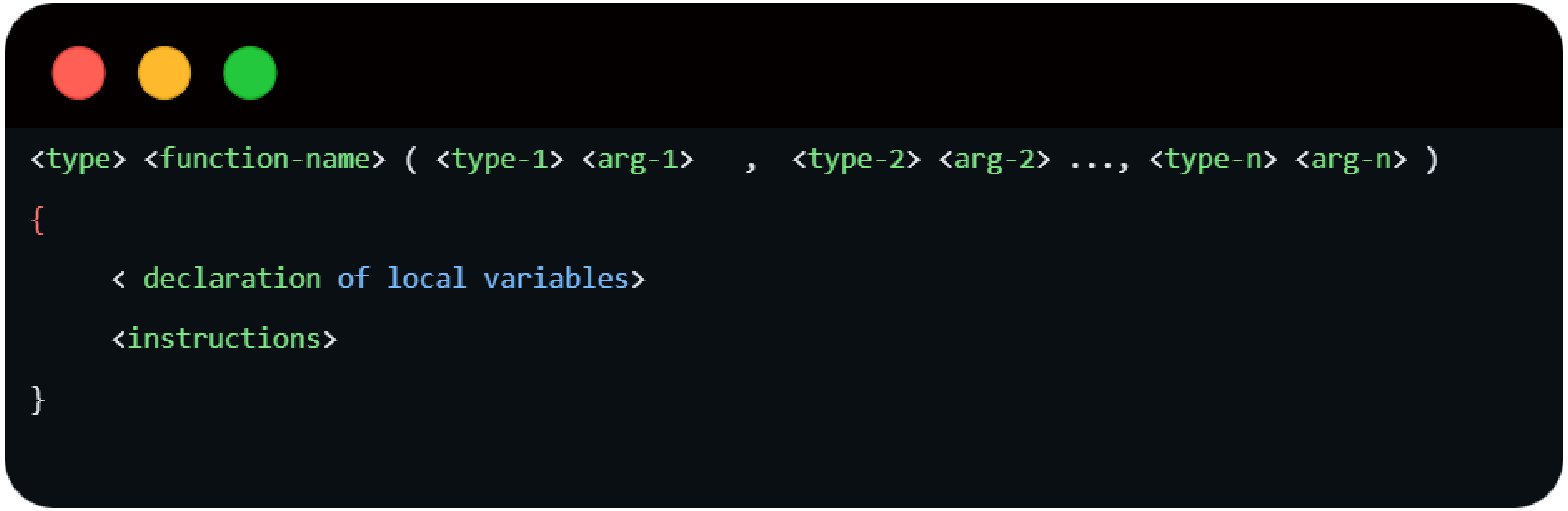
BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

- **Definition of a Function in C**

It is a sub-program which performs a specific task, its definition includes a header where the name of the function is specified, the type of result that the function returns and the parameters of the function, a declaration part of local variables used by the function and an instruction part called the body of the function. It is of the form:



```
<type> <function-name> ( <type-1> <arg-1>   ,  <type-2> <arg-2> ..., <type-n> <arg-n> )
{
     < declaration of local variables>
     <instructions>
}
```

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

<type> is the type of the value returned by the function. In C there is no concept of procedure. If a function does not return a result, we specify the type using the void keyword .

<arg-1>, <arg-2> ... <arg-n> are the formal parameters of the function.
<type-1>, <type-2> ... <type-n> are the successive types of these parameters.
If the function has no parameters, we replace the list of formal parameters with the keyword void or simply put nothing in the list of parameters.

The function body optionally begins with variable declarations, which are local to this function. It ends with the return statement to the calling function, return, whose syntax is “ return(expression); ”.
The value of expression is the value that the function returns. Its type must be the same as that which was specified in <type>. If the function does not return a value (void type function), its definition ends with “ return; ” or we can not put “ return; " in this case.

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

Remarks :

· A function can provide as result:

- an arithmetic type (integer, real, character or enumeration),
- a structure (defined by struct : we will see the structs later on),
- a pointer (we will see pointers later)
- void (the function then corresponds to a "procedure ").

· A function cannot return arrays, strings or functions as results. However, it is possible to return a pointer to the first element of an array or a string or a pointer to a function.

· You cannot define functions inside another function (like in Pascal) in C.

· main() is the main function in C where the program begins execution.

suivant →

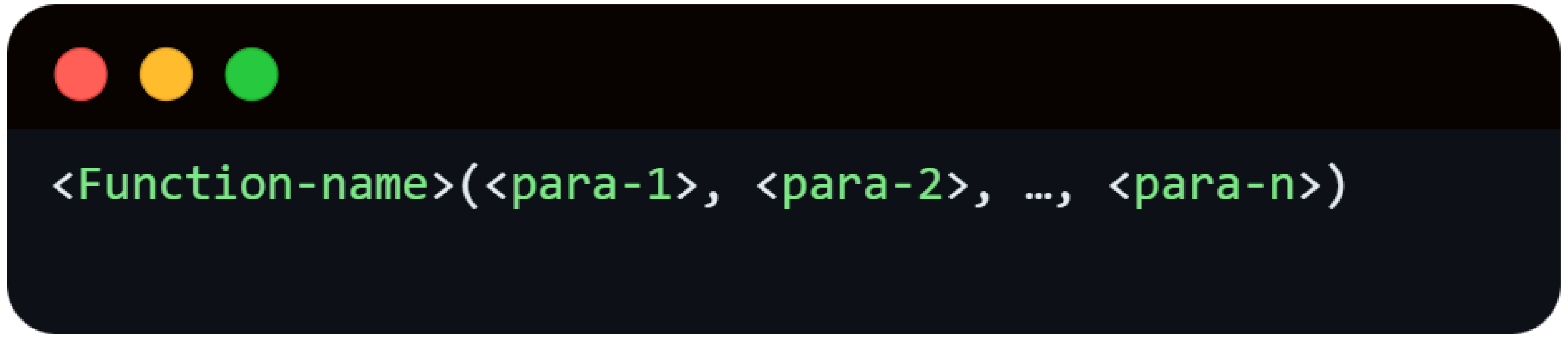
BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

Calling a Function

A function is called using the expression:



```
<Function-name>(<para-1>, <para-2>, …, <para-n>)
```

The order and type of the function's real parameters (arguments) must match those given in the function definition.

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

Example :

```
 #include <stdio.h>
int  sum2Integers(int a, int b)
{
    int s ;
   s=a+b;
     return s;
}
int main()
{
   int x, y, sum;
   printf( "Enter x and y 2 integers \n");
  scanf("%d %d", &x, &y);
   sum = sum2Integers(x, y);
   printf("The sum of x and y = %d\n", sum);

  return 0;
 }
```

suivant →

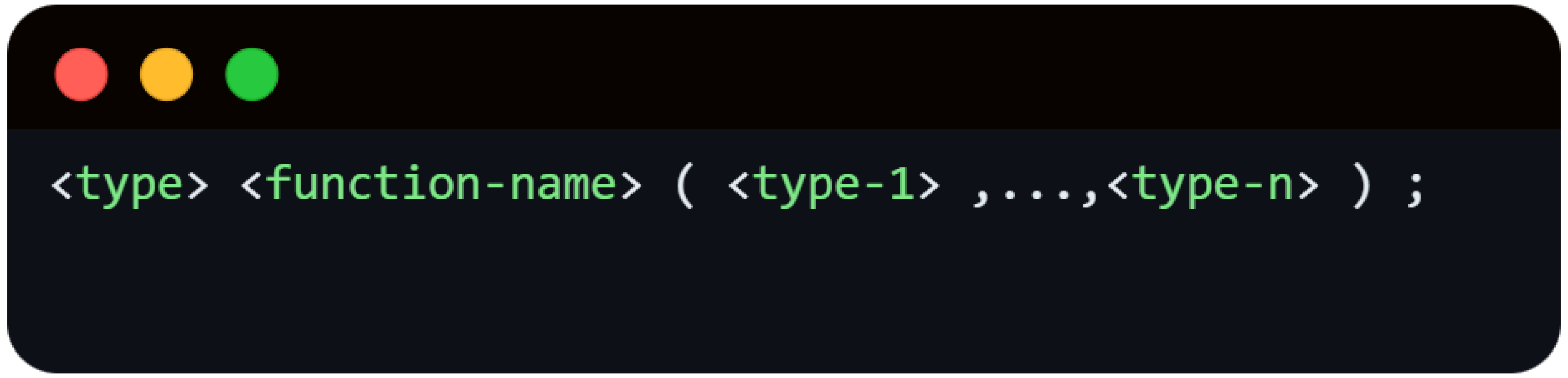
BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

Prototype of a Function :
 The definition of a secondary function can be placed either before or after the main function main(). However, it is essential that the compiler “knows” the function when it is called.

If a function is defined after its first call (in particular if its definition is placed after the main function), it must be declared beforehand. A secondary function is declared by its prototype , which gives the type of the function and that of its parameters, in the form:



```
<type> <function-name> ( <type-1> ,...,<type-n> ) ;
```

If in the previous example the function sum2Integers() would have been defined after main(), we will have to add the prototype of this function as shown below.

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

Example :

```
#include <stdio.h>

int sum2Integers(int , int);

int main()
{
   int x, y, sum;
   printf("Enter x and y 2 integers \n");
   scanf("%d %d", &x, &y);
   sum = sum2Integers(x, y);
   printf("The sum of x and y = %d\n", sum);

  return 0;
 }
int sum2Integers(int a, int b)
{
    int s ;
   s=a+b;
     return s; }
```

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

Global Variables and Local Variables

- **Global Variables**

A variable declared outside of any function is called a global variable . A global variable is known to the compiler throughout the portion of code following its declaration. Global variables are always permanent. They are created before the program begins its execution and exist until the end of the program execution.

In the following program, glob is a global variable:

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

```
int glob;

void f()
{
   glob++;
   printf("glob = %d\n", glob);
   return;
}
int  main()
{
  int i;
   glob = 0 ;
   for (i = 0; i < 5; i++)
      f();

  return 0;
}
```

In fact, the program displays

```
glob = 1
glob = 2
glob = 3
glob = 4
glob = 5
```

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

- **Local Variables**

A local variable is a variable declared inside a function (or a block of instructions) of the program. By default, local variables are temporary. When a function is called, it places its local variables on the stack. When the function exits, the local variables are unstacked and therefore lost. Variables local to a function have a lifetime limited to a single execution of this function. Their values are not preserved from one call to the next.

If we declare a local variable with the same name as a global variable, the variable referenced in the function where the local variable is declared is the local variable.
For example the following program:

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

```
int n = 10;
void f()
{
   int n = 0;
   n++;
   printf("n = %d\n",n);
   return;
}
int main()
{
   int i;
   for (i = 0; i < 5; i++)
       f();
  return 0;
}
```

displays:

```
n = 1
n = 1
n = 1
n = 1
n = 1
```

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

Passing Parameters to a Function

The parameters of a function are treated in the same way as local autoclass variables when calling the function, the effective parameters are copied to the stack segment. The function then works only on these copies. These copies disappear when returning to the calling program.

Passing Parameter by Value

When a variable is passed by value to a function, a copy of the value is put into the function's stack segment. In particular if the called function modifies this value it modifies the copy which is on the stack and the original value in the calling function is not modified.

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Functions

Passing an Array as a Parameter to a Function

In C, an Array is always passed by address as a parameter to a function.

For example the program:

```
void init (int tab[], int n)
{
  int i;
  for (i = 0; i < n; i++)
      tab[i] = i;
  return;
}
void displayVect(int V[], int n)
{
  int i;
  printf("Elements of Vector V:\n");
  for (i = 0; i < n; i++)
      printf("V[%d] = %d\n", i, V[i]);
  return;
}
```

```
int main()
{
 int I;
 int V[5];

 init(tab,n);
 displayVect(V, 5);

 return 0;
}
```

Displays:

```
Elements of Vector V:
V[0] = 0
V[1] = 1
V[2] = 2
V[3] = 3
V[4] = 4
```

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Application 1

Write the algorithm for determining the GCD of two integers A and B using Euclidean division.

We use the properties:
GCD(a, 0) = a
GCD(a, b) = GCD(b, a mod b) for b # 0

Euclid's algorithm consists of performing a series of Euclidean divisions:
- We carry out the Euclidean division of a by b and we write the remainder by r.
- Then, b becomes a and r becomes b as in the table below; and we start again: we carry out the Euclidean division of a by b and we write the remainder by r.
- And we continue like this until a division gives a remainder equal to 0. .

PGCD (7038, 5474)

A	B	Rest
7038	5474	1564
5474	1564	782
1564	782	0
782	0	

GCD(7038, 5474) = 782

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

Application 2

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Write a Square function checking if a natural number is a perfect square (using only the basic operators).
Hint: X is a perfect square if there exists an integer i such that X = i * i.

Write a program that asks the user to enter a positive integer N and displays the list of perfect squares from 1 to N. The program will ask the user again to enter a new N if they wish to continue.

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

Application 3

ADRESSE EMAIL
beldjouziasmaa@gmail.com

a) Write a function that returns True if the character passed as a parameter is equal to ¢ o ¢ or ¢ O ¢ (which means Yes), and False otherwise.

b) Write an algorithm that asks the user for an integer and then displays the multiplication table from 1 to 9 of that number. You should use the previous function to allow the user to provide numbers for as long as they want.

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

Application 3

ADRESSE EMAIL
beldjouziasmaa@gmail.com

a) Write a function that returns True if the character passed as a parameter is equal to ¢ o ¢ or ¢ O ¢ (which means Yes), and False otherwise.

b) Write an algorithm that asks the user for an integer and then displays the multiplication table from 1 to 9 of that number. You should use the previous function to allow the user to provide numbers for as long as they want.

suivant →

BELDJOUZI ASMAA
ingénieure en informatique

Application 4

ADRESSE EMAIL
beldjouziasmaa@gmail.com

Write an algorithm displaying all perfect numbers less than 200. Knowing that a positive integer (N) is perfect if it is equal to the sum of its divisors (<N). We will write a Boolean function, called PERFECT, taking an integer, and returning true if the number is perfect, false otherwise.
Examples: 6 — which is equal to 1 + 2 + 3 —
and 28 — which is equal to 1 + 2 + 4 + 7 + 14 — are perfect numbers.