Cours 3:

Basic Concepts of Programming in C

Dr. Chénche

"Programming", indeed! But which language should we use?

Evidence : Computers only understand binary (0 and 1),

Problem: but writing everything in binary would be impossible for humans.

Solution: To make communication simpler, people created several programming languages:

like C, C++, Java, Visual Basic, Python, etc.

What Do You Need for C Programming?

- Programmers need three tools: a text editor, a compiler, and a debugger.
- You can install these tools separately or use an Integrated Development Environment (IDE).
- IDEs like Code::Blocks, Visual C++, Dev-C++.

1. My First C Program

Example C Program

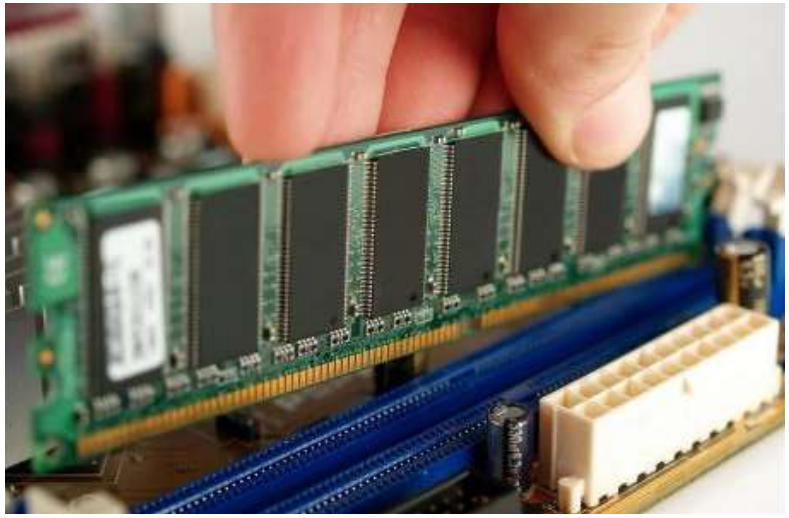
```
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

- When you run this program, it displays on the screen **Hello, World!** and then terminate with a **success status of 0.**

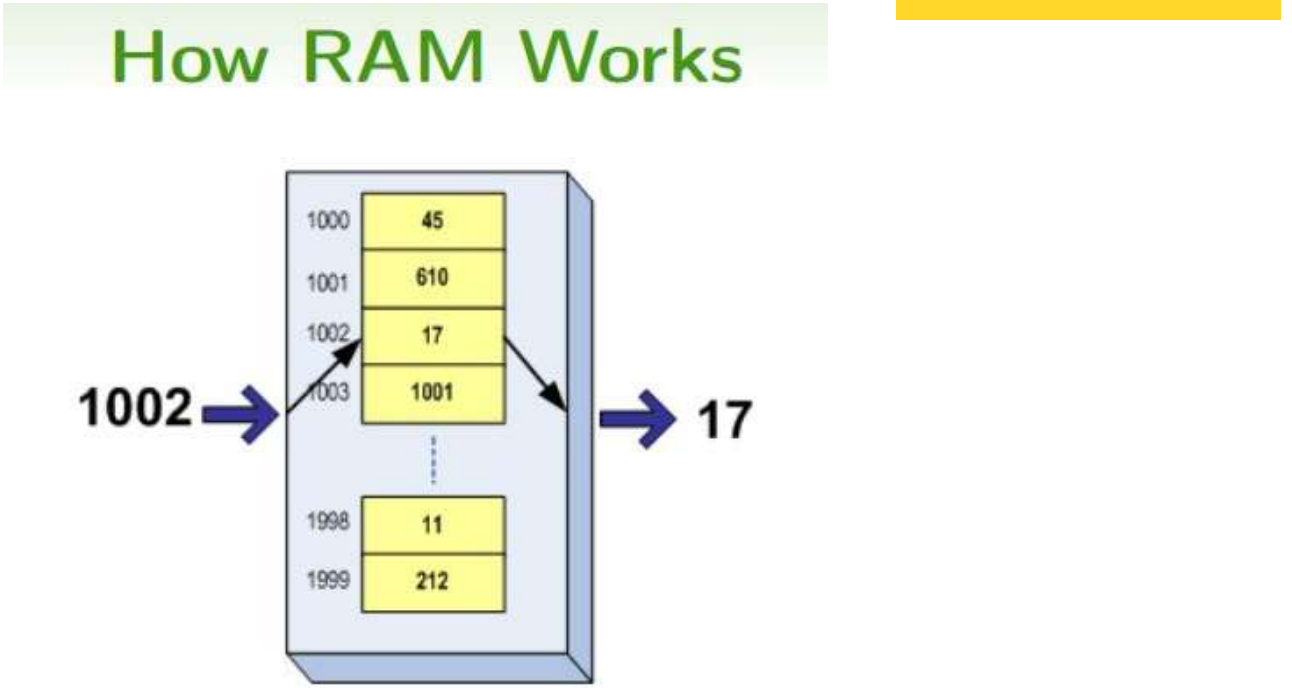
2. Variables and Constants :

Problem : Write a program that calculates the sum of two integers.

- The computer needs to remember the two integers before performing the addition operation.
- Therefore, it needs memory!



2. Variables and Constantes :



1. **Addresses:** An address is a number that allows the computer to identify the location of a specific cell in RAM.
2. **Values:** At each address, you can store a value (a number). You can only store one number per address!

2. 1. Variable Concept :

- A variable is temporary information that we store in memory.
- It is called a "**variable**" because it is a value that can change during program execution.
- In the C language, a variable consists of three things:

  1. An identifier (name) : this is what allows us to recognize it.

  2. A value: this is the number it stores, for example, 7;

  3. A type: what kind of data it can hold.

a. Variable identifier :

In C programming language, there are some constraints to follow when naming variables:

- You can only use lowercase and uppercase letters and numbers (abcABC012);
- Spaces are not allowed. Instead, you can use the underscore character _.
- Your variable name must start with a letter or _ ;
- You are not allowed to use accents (éàê, etc.).

Example Which variable names are incorrect:

_age, 2mail, name_2, score2, my book, phone_number, prénom, main.

a. Variable identifier :

- Each programmer may have their own style in nomming variables, but it's preferable to :
- Don't use only basic alphabets x,y, a,b .
- Choose **meaningful, descriptive names** instead of short, unclear abbreviations (age, time).
- For multi-word names, use **camelCase** (e.g., bookTitle, stdt_Age).

b. Variable Types

- There are several types of numbers: 327, 47.10, -38, 68597.00007654
- When you create a variable, you must indicate its type,
- Here are the main types of variables in the C language:

b. Variable Types

Type	Meaning	Typical Size*	Example Declaration
int	Integer	4 bytes	int a = 10;
short	Short integer	2 bytes	short s = 100;
long	Long integer	4 or 8 bytes	long l = 100000;
unsigned int	Integer (no negative values)	4 bytes	unsigned int u = 50;
float	Real number (decimal, single precision)	4 bytes	float x = 3.14;
double	Real number (double precision)	8 bytes	double pi = 3.141592653;
char	Character	1 byte	char c = 'A';
bool	Boolean (true/false)	1 byte	bool b = 1;

b. Variable Types

- For an integer, you will most often use **int**.
- For a real (or decimal) number, you will use **double** or **float**.

c. Declaring a Variable

- **You must declare variables at the beginning of functions. Just do the following:**

1. Specify the type of the variable you want to create;
2. Insert a space;
3. Specify the name you want to give to the variable;
4. Finally, don't forget the semicolon.

Syntax :

```
Type_of_var identifier;
```

Eg.

```
int x; float y; char z;char S[50];
```

d. Initializing a Variable

- **When you declare a variable, what value does it have at the beginning?**
- Initializing a variable means combining the declaration and assignment of that variable in the same statement .
- The advantage is that you are sure that this variable contains a correct value, not just anything.

Syntax :

```
Type_of_var identifier = initial_value;
```

d. Initializing a Variable

Example:

```
#include <stdio.h>

int main()
{
   int studentNumbers = 240;

   return 0;
}
```

2.2. Constants :

- A constant is a **variable** whose value **does not change** during the **execution** of the **program**;
- it can be an integer or real number, a character.
- In C, a constant is declared as follows

Syntax :

```
Const Type identifier = value;
```

Question : Give the declaration of all the *variables and constants* necessary to *calculate the area of a circle*.

3. ASSIGNMENT :

- In C, assignment refers to the operation of giving a value to an existing variable.
- This happens through the operator **=**.
- In C, the sign **=** is the assignment operator, not the comparison operator.
- **Examples :**

Simple assignment :

```
int a; // Declaration of variable a
a=5; // Assigns the value 5 to a
```

3. ASSIGNMENT :

Assignment with expression :

```
int a , b;
a = 25;
b= a + 2; // Assigns to b the value of a+2
```

Assignment by copy :

```
int a,b;
a=5;
b=10;
a=b; // a receives a copy of b
```

4. Operators :

4.1. Mathmatical operators in C

Operator	Meaning	Example	Result
+	Addition	5 + 3	8
-	Subtraction	5-3	2
*	Multiplication	5 * 3	15
/	Division	7/2	3
%	Modulo (remainder)	7 % 2	1

4.2. Logical Operators in C

Operator	Meaning	Example	Result
!	NOT	!(5 == 5)	false
&&	AND	(5 > 3 && 2 < 0)	false
||	OR	(5 > 3 || 2 < 0)	true

4. Operators :

4. 3. Relational Operators in C

Operator	Meaning	Example	Result
==	Equal to	5 == 5	true
!=	Not equal to	5 != 5	false
>	Greater than	7 > 3	true
<	Less than	2.5 < 0	false
>=	Greater than or equal to	5 >= 5	true
<=	Less than or equal to	3 <= 5	true

4. Operators :

4.4. Compound Assignment Operators

C also provides **shorthand operators** that combine arithmetic with assignment : **+=, -=, *=, /= %=.**

These make the code shorter and easier to read.

Operator	Meaning	Example	Equivalent to
+=	Add and assign	x += 5;	x = x + 5;
-=	Subtract and assign	x -= 3;	x = x - 3;
*=	Multiply and assign	x *= 2;	x = x * 2;
/=	Divide and assign	x /= 4;	x = x / 4;
%=	Modulo and signe	x /=3;	x = x % 3;

4. Operators :

4.5. Increment/decrement in C

Operators The increment (**++**) and decrement (**--**) operators provide a convenient way of, respectively, **adding** and **subtracting 1** from a numeric variable. These are summarized in the following table.

4. Operators :

4.5. Increment/decrement in C

Operator	Name	Example	(For x = 5)	Explanation
++x	Pre-increment	y = ++x;	x = 6, y = 6	x is increased before being used
x++	Post-increment	y = x++;	x = 6, y = 5	x is increased after being used
--x	Pre-decrement	y = --x;	x = 4, y = 4	x is decreased before being used
x--	Post-decrement	y = x--;	x = 4, y = 5	x is decreased after being used

4. Operators :

4.6. Precedence of operators

Precedence	Operator(s)	Associativity
1	()	Left to right
2	functions	Left to right
3	! , unary -	Right to left
4	*, /, %	Left to right
5	+, -	Left to right
6	<, <=, >, >=	Left to right
7	==, !=	Left to right
8	&&	Left to right
9	||	Left to right

5. Common functions from <math.h> in C

Function	Description	Example
floor(x)	Floor = partie entière (arrondit vers le bas)	floor(3.7) → 3.0
fabs(x)	Valeur absolue (float absolute)	fabs(-5.2) → 5.2
sqrt(x)	Racine carrée	sqrt(16) → 4.0
pow(x, y)	Puissance (x^y)	pow(2, 3) → 8.0
exp(x)	Exponentielle (e^x)	exp(1) → 2.718...
log(x)	Logarithme népérien (base e)	log(2.718) → 1.0
log10(x)	Logarithme base 10	log10(1000) → 3.0
sin(x)	Sinus (x en radians)	sin(3.14159/2) → 1.0
cos(x)	Cosinus	cos(0) → 1.0
tan(x)	Tangente	tan(3.14159/4) → 1.0

6. Displaying in C :

- We use **printf** in a similar way to display **text,**
- If we want display the **variable's value** we add special symbols. For example:

```
int main ()
{
        int studentNumbers = 240;
        printf ("Il y a %d etudiants inscrits .",
                studentNumbers );
```

- The letter after **%** indicates what should be displayed. **'d'** means we want to display an **int**.

6. Displaying in C :

Common printf Format Specifiers :

Specifier	Expected Data Type	Example Code	Output
%d / %i	int (signed integer)	printf("%d", 42);	42
%u	unsigned int	printf("%u", 42);	42
%f	float / double (decimal number)	printf("%f", 3.14159);	3.141590
%2.3f	float/double with width and precision of decimal places	printf("%.2f",3.14159);	3.14
%c	char (single character)	printf("%c", 'A');	A
%s	string (char array ending with \0)	printf("%s", "Hello");	Hello

6. Displaying in C :

Sytax of printf :

printf (% [Width] [.precision] [spesifier]) ;

With printf we can display :

- **Litteral (texte) eg. printf(" Hello ");**
- **Variable/Constant eg. Printf (" %3.4f", x);**
- **Value eg. Printf (" %d", 5);**
- **Expression eg. Printf (" %d", X+Y);**
- **Escape sequences eg. Printf (" \n");**

6. Displaying in C :

Escape sequences :

Escape sequence	Meaning	Example	Output
\n	New line	printf("Hello \n World");	Hello World
\t	Horizontal tab	printf("Hello \t World");	Hello World
\\	Backslash (\)	printf("\\");	\
\"	Double quote (")	printf("\"Hello\"");	"Hello"

6. Displaying in C :

Displaying Multiple Variables with a Single printf

It is possible to display the values of multiple variables in a single printf. To do this, you need to specify **%d** or **%f** where you want, and then specify the corresponding variables in the same order, separated by commas.

6. Displaying in C :

Displaying Multiple Variables with a Single printf

```
int  main ()
{
        int  studentNumbers  =  240;
        double  average  =  14.5  ;

        printf ("Il  y  a  %d  etudiants  inscrits  avec  une
              moyenne  de  %f  en  Bac",  studentNumbers ,
              moyenneBac );

        return  0;
}
```

6. Displaying in C :

```
#include <stdio.h>
#include <math.h>   // pour pow()
int main() {
    int entier = 42;
    float reel = 23.1415;
    char caractere = 'A';
    char chaine[50] = "Hello"; // string
    const int define = 100;
printf("Section A Autonomuous System \n");
    printf("Entier : %10i\n", entier);
    printf("Reel (float) : %10.3f\n", reel);
    printf("Caractere : %c\n", caractere);
    printf("Chaine : %s\n", chaine);
    printf("Constante (define) : %d\n", define);
    printf("Valeur directe : %d\n", 25);
    printf("Expression 3 + 5 = %d\n", 3 + 5);
    printf("2 puissance 3 = %1.2f\n", pow(2, 3));
    printf("Texte avec tabulation Bonjour \t Monde\n");
    printf("Texte avec guillemets -> \"Bonjour\"\n");
    printf("Resultat logique (5 > 2) = %d\n", (5 > 2));
    return 0;
}
```

7. Reading in C :

Ask the user to enter a value of a variable

• We use another ready-made function called **scanf**.

• This function is similar to printf. You must specify a format to indicate what the user needs to enter (int, float, etc.)

• Then you must specify the name of the variable that will receive the number.

7. Reading in C :

Ask the user to enter a value of a variable

```
int main ()
{
        int age = 0;
        scanf ("%d", &age );

        return 0;
}
```

%d must be enclosed in quotes.

- Furthermore, you must put the **&** symbol in front of the variable name that will receive the value.

7. Reading in C :

Important :

- **Don’t put the & in front of string variable**
- ***To read a char you can also use :***

  identifier=getchar();

- ***To read a string you can also use :***

  fgets(identifier,sizeof(idfentifier),stdin);

1. My First C Program

Comments in C

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

/* Below, you have the main function of the program, called
    main. All programs start with this function. Here, my
    function simply displays "Hello" on the screen. */

int main()
{
  printf("Hello"); // This instruction displays "Hello" on
       the screen
  return 0;    // The program returns the number 0 and stops
}
```

8. Exercice :

Write a simple program that asks the user's age and then displays it:

```
int  main ()
{
   int  age  =  0;  // We  initialize  the  variable  to  0

   printf (" How  old  are  you ?");
   scanf ("%d", & age );  // We  ask  to  enter  the  age  with  scanf
   printf ("Ah!  So  you  are  %d  years  old !\n\n",  age );"

   return  0;
}
```