Variable

Question: 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!

Variable

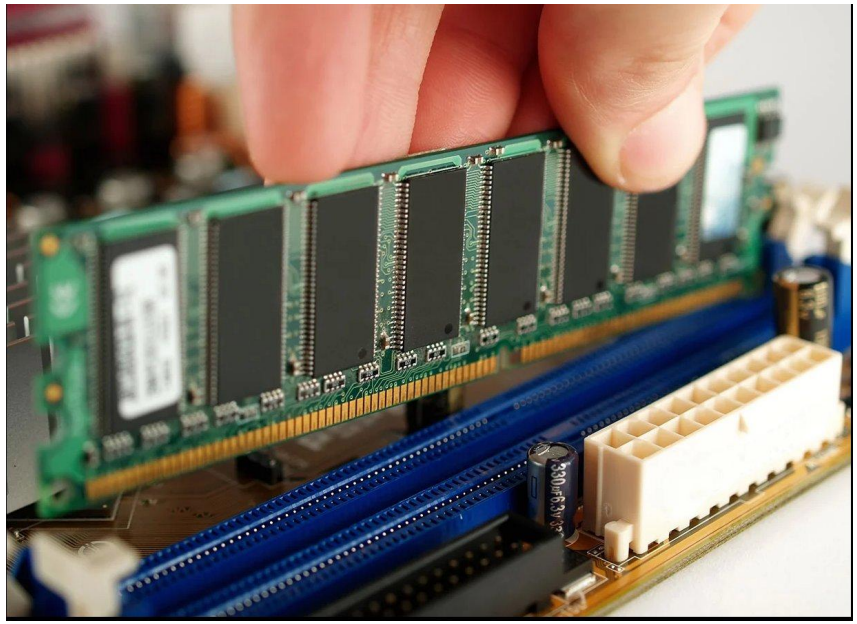
Question: Write a program that calculates the sum of two integers.

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

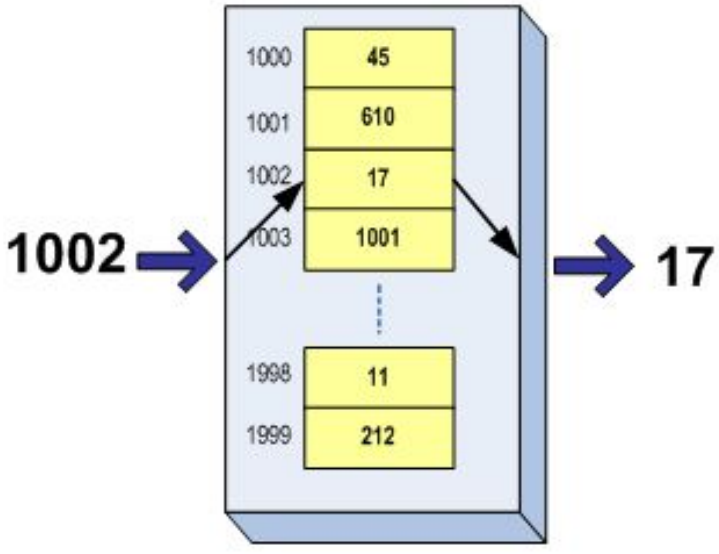
Variable

Question: 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!



How RAM Works



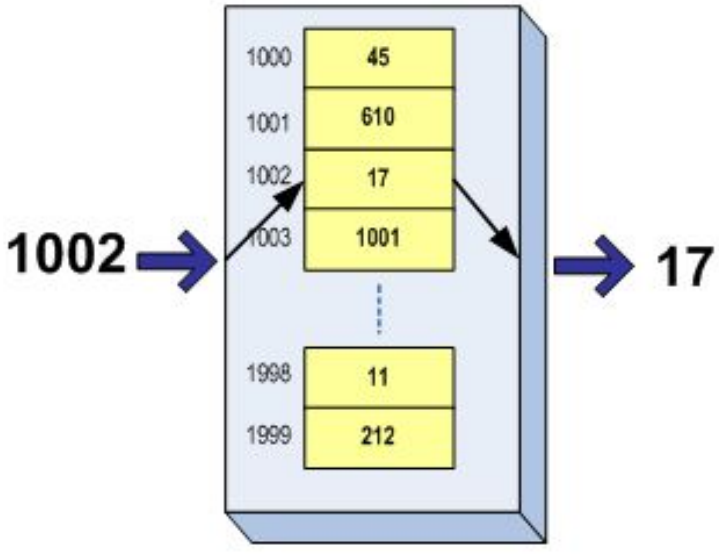
1. **Addresses:** An address is a number that allows the computer to locate itself in RAM.

can only store one number per address!

Problem

example?

How RAM Works



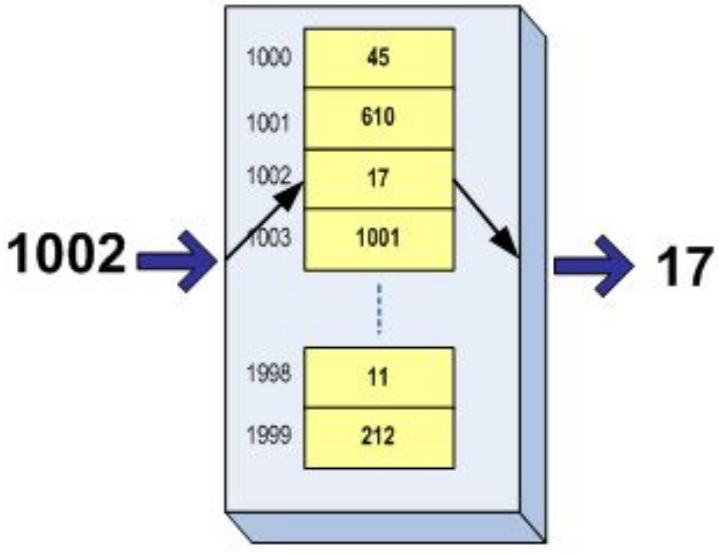
1. **Addresses:** An address is a number that allows the computer to locate itself in RAM.

can only store one number per address!

Problem

example?

How RAM Works



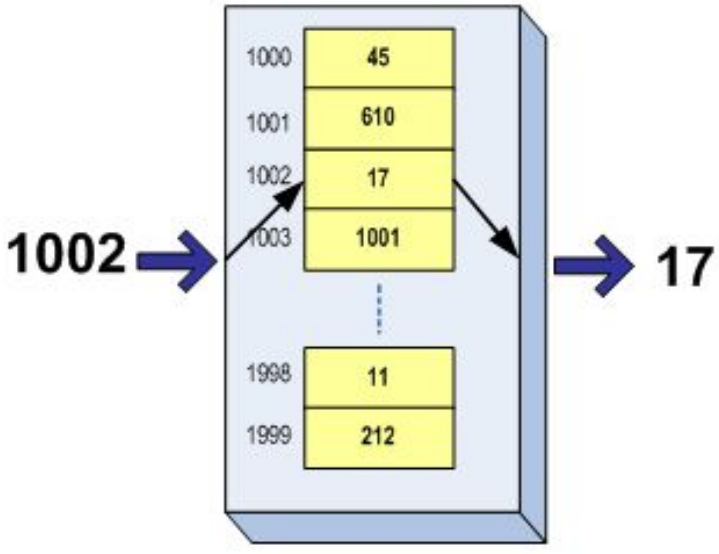
1. **Addresses:** An address is a number that allows the computer to locate itself in RAM.

   can only store one number per address!

Problem

example?

How RAM Works



1. **Addresses:** An address is a number that allows the computer to locate itself in RAM.

   can only store one number per address!

Problem

example?

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 two things:

1. **a name:** this is what allows us to recognize it. In programming, we don't need to remember the memory address; we just indicate

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

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 two things:

1. **a name:** this is what allows us to recognize it. In programming, we don't need to remember the memory address; we just indicate

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

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 two things:

1. **a name:** this is what allows us to recognize it. In programming, we don't need to remember the memory address; we just indicate
2. **a value:** this is the number it stores, for example, 7;

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 two things:

1. **a name:** this is what allows us to recognize it. In programming, we don’t need to remember the memory address; we just indicate

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

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 two things:

1. **a name:** this is what allows us to recognize it. In programming, we don’t need to remember the memory address; we just indicate

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

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

- You can only use lowercase and uppercase letters and numbers (abcABC012);

- Your variable name must start with a letter;
- Spaces are not allowed. Instead, you can use the underscore character _. It is the only character different from letters and numbers that is allowed;

- You are not allowed to use accents (éàê, etc.).

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Naming Variables

In most programming languages, there are some constraints and conventions to follow when naming variables:

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

Example

phone_number, prénom, main.

Some Good Practices for Naming Your Variables

- Every programmer has his way of naming variables;
- Start all variable names with a lowercase letter;
- If there are multiple words in the variable name, capitalize the first letter of each new word. For example: bookTitle or lastChangedParameter
- Make sure to give your variables meaningful and descriptive names. While it might be tempting to shorten "lastChangedParameter" to "lcp" for brevity, this abbreviation can obscure the variable's purpose when reviewing your code. Therefore, don't hesitate to use slightly

Some Good Practices for Naming Your Variables

- Every programmer has his way of naming variables;
- Start all variable names with a lowercase letter;
- If there are multiple words in the variable name, capitalize the first letter of each new word. For example: bookTitle or lastChangedParameter
- Make sure to give your variables meaningful and descriptive names. While it might be tempting to shorten "lastChangedParameter" to "lcp" for brevity, this abbreviation can obscure the variable's purpose when reviewing your code. Therefore, don't hesitate to use slightly

Some Good Practices for Naming Your Variables

- Every programmer has his way of naming variables;
- Start all variable names with a lowercase letter;
- If there are multiple words in the variable name, capitalize the first letter of each new word. For example: bookTitle or lastChangedParameter
- Make sure to give your variables meaningful and descriptive names. While it might be tempting to shorten "lastChangedParameter" to "lcp" for brevity, this abbreviation can obscure the variable's purpose when reviewing your code. Therefore, don't hesitate to use slightly

Some Good Practices for Naming Your Variables

- Every programmer has his way of naming variables;
- Start all variable names with a lowercase letter;
- If there are multiple words in the variable name, capitalize the first letter of each new word. For example: bookTitle or lastChangedParameter
- Make sure to give your variables meaningful and descriptive names. While it might be tempting to shorten "lastChangedParameter" to "lcp" for brevity, this abbreviation can obscure the variable's purpose when reviewing your code. Therefore, don't hesitate to use slightly

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:

1. For an **integer**, you will most often use **int**;

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:

1. For an **integer**, you will most often use **int**;

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:

1. For an **integer**, you will most often use **int**;

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:

Nom du type	Minimum	Maximum
signed char	-127	127
int	-32 767	32 767
long	-2 147 483 647	2 147 483 647
float	$3.4*10^{-38}$	$3.4*10^{38}$
double	$1.7*10^{-308}$	$1.7*10^{308}$

1. For an **integer**, you will most often use **int**;

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:

Nom du type	Minimum	Maximum
signed char	-127	127
int	-32 767	32 767
long	-2 147 483 647	2 147 483 647
float	$3.4*10^{-38}$	$3.4*10^{38}$
double	$1.7*10^{-308}$	$1.7*10^{308}$

1. For an **integer**, you will most often use **int**;

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:

Nom du type	Minimum	Maximum
signed char	-127	127
int	-32 767	32 767
long	-2 147 483 647	2 147 483 647
float	$3.4*10^{-38}$	$3.4*10^{38}$
double	$1.7*10^{-308}$	$1.7*10^{308}$

1. For an **integer**, you will most often use **int**;

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;

4. Finally, don't forget the semicolon.

```
int main ()
{
int age ;
double salary ;
unsigned int sum , studentNumber , coefficient ;

return 0;
}
```

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;

4. Finally, don't forget the semicolon.

```
int main ()
{
int age ;
double salary ;
unsigned int sum , studentNumber , coefficient ;

return 0;
}
```

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;

4. Finally, don't forget the semicolon.

```
int main ()
{
int age ;
double salary ;
unsigned int sum , studentNumber , coefficient ;

return 0;
}
```

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;

4. Finally, don't forget the semicolon.

```
int main ()
{
int age ;
double salary ;
unsigned int sum , studentNumber , coefficient ;

return 0;
}
```

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;

4. Finally, don't forget the semicolon.

```
int main ()
{
int age ;
double salary ;
unsigned int sum , studentNumber , coefficient ;

return 0;
}
```

Assigning a Value to a Variable

Simply specify the variable name, then an equal sign (=), and finally the value you want to put there.

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

int   main ()
{
    int   studentNumbers ;
    studentNumbers   =   240;

    return   0;
}
```

Initializing a Variable

- When you declare a variable, what value does it have at the beginning?

assignment of that variable in the same statement

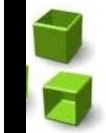
correct value, not just anything.

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

int   main ()
{
    int   studentNumbers   =   240;

    studentNumbers   =   260
    studentNumbers   =   255
    studentNumbers   =   258

    return   0;
}
```



Initializing a Variable

- When you declare a variable, what value does it have at the beginning?

  assignment of that variable in the same statement

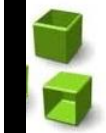
  correct value, not just anything.

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

int   main ()
{
    int   studentNumbers   =   240;

    studentNumbers   =   260
    studentNumbers   =   255
    studentNumbers   =   258

    return   0;
}
```



Initializing a Variable

- When you declare a variable, what value does it have at the beginning?

  assignment of that variable in the same statement

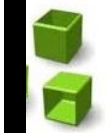
  correct value, not just anything.

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

int   main ()
{
    int   studentNumbers   =   240;

    studentNumbers   =   260
    studentNumbers   =   255
    studentNumbers   =   258

    return   0;
}
```



Initializing a Variable

- When you declare a variable, what value does it have at the beginning?

  assignment of that variable in the same statement

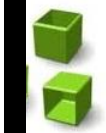
  correct value, not just anything.

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

int   main ()
{
    int   studentNumbers   =   240;

    studentNumbers   =   260
    studentNumbers   =   255
    studentNumbers   =   258

    return   0;
}
```



Displaying a Variable

We use **printf** in a similar way to display text, except that we add a special symbol where we want to display the variable's value. 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**.

Displaying a Variable

We use **printf** in a similar way to display text, except that we add a special symbol where we want to display the variable's value. 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**.

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.

```
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;
}
```

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.

```
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.

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.

```
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.

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.

```
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.

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.

```
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.

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.

```
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.

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.

```
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.

A Little Example to Conclude

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;
}
```

Simple Addition Calculator Program

```
int main ()
{
    int result = 0, number1 = 0, number2 = 0;

    // We ask the user for numbers 1 and 2:
    printf (" Enter number 1: ");
    scanf ("%d", & number1 );
    printf (" Enter number 2: ");
    scanf ("%d", & number2 );

    // We perform the calculation :
    result = number1 + number2 ;

    // And we display the addition on the screen :
    printf ("%d + %d = %d\n", number1 , number2 , result );

    return 0;
}
```