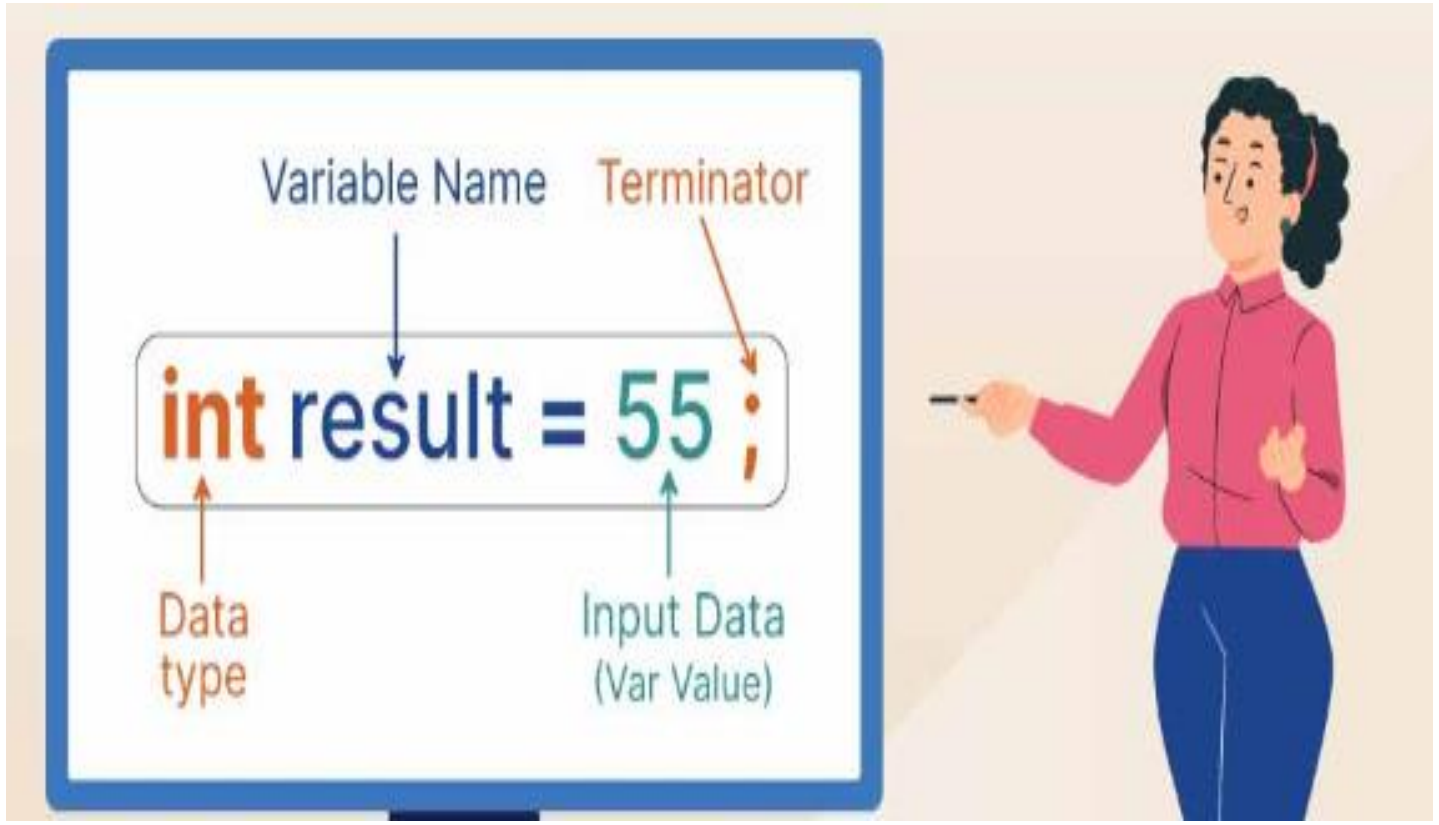
WEEK 4 and 5: VARIABLES, STATEMENTS &EXPRESSIONS

By

Michael Kumakech

VARIABLES

- Variables in C Programming are a **container** to hold the value of a **data type**.
- A variable is a memory allocation space to a data type.
- They help the compiler to create memory space for the type of Variable.
- Each variable has a unique name.



Syntax of Variable in C

Syntax of Variable in C is:

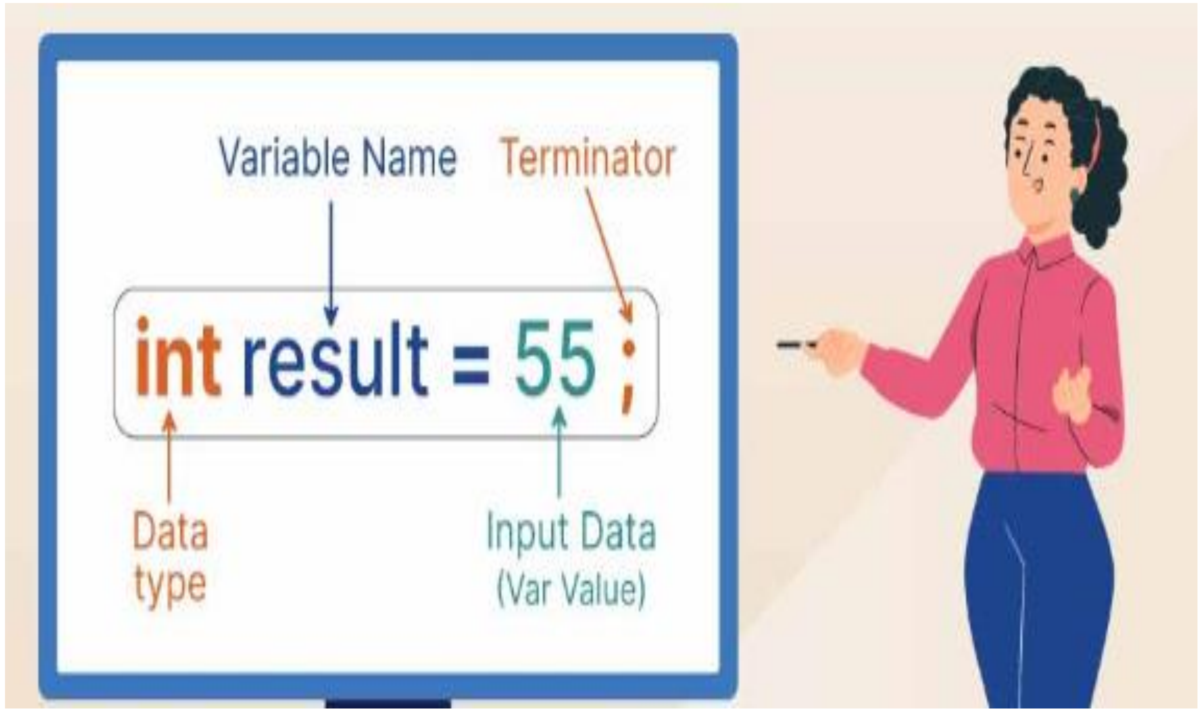
data_type variable_name = value

Example:

- ***int num = 3***

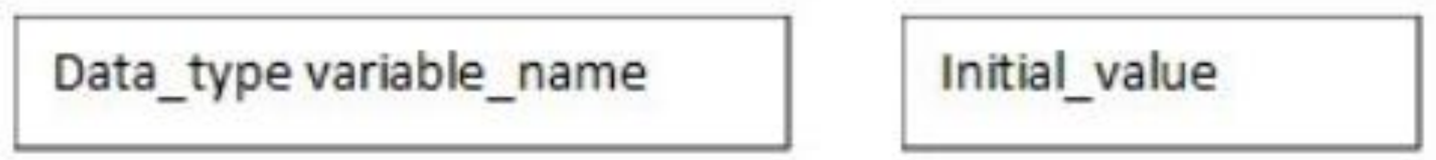
Here,

- **int** is the data type
- **num** is a variable name
- **3** is the variable value



What is a Variable in C Programming

- ***A Variable in C programming consists of two parts, Variable definition, and initialization.***
- **Variable definition** = it consists of the data type and the variable name. This part helps to identify the variable data type and its identifier. Each data type consumes different memory, so it is necessary to define the variable type.
- **Variable initialization** = variable initialization means to provide some value to the variable as per the data type.



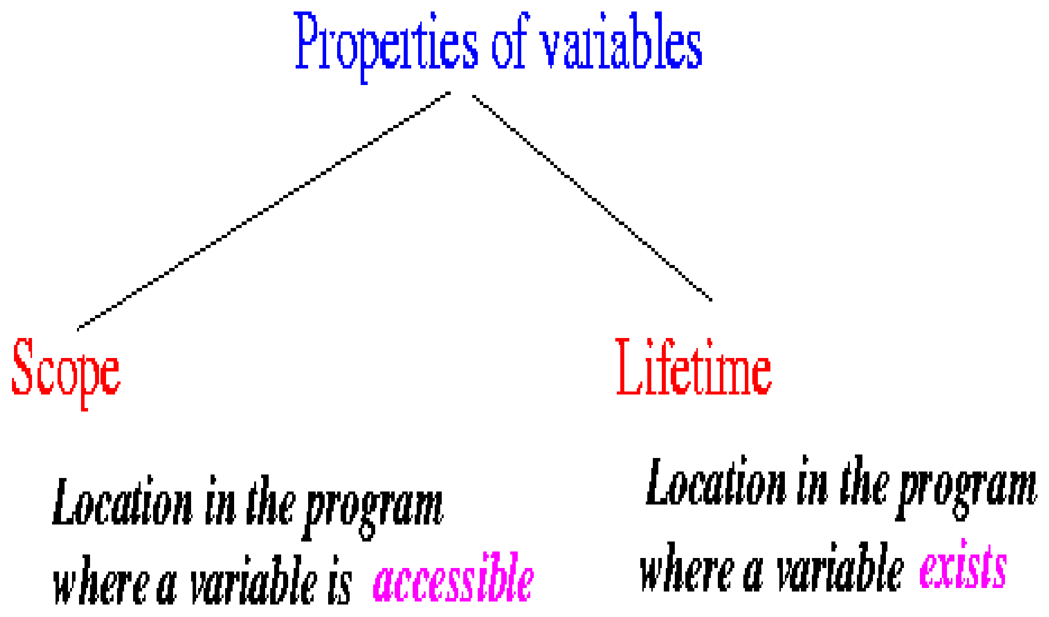
Variable definition Variable Initialization

Rules for Variable Names in C Programming

1. Variable names will always start with an alphabet and underscore. Example: **num, name, a, x, _value.**
2. Variable names in C will never start with a digit. Example: 4name, is an invalid name.
3. Variables should be declared with their data type. Variable names without data type will generate errors. For example, **int a =2 is valid**. **a = 2 is an invalid** variable.
4. C is a strongly typed language, you cannot change the data type of a variable after its definition.
5. Reserved **keywords** cannot be used as a variable name in C programming.
6. No **white spaces** are allowed within the variable name.

What is the Scope and Lifetime of a Variable?

- The scope is the variable region in which it can be used. Beyond that area, you cannot use a variable.
- Global scope is the entire program. Global variables can be used anywhere throughout the program.
- Lifetime is the life or alive state of a variable in the memory. It is the time for which a variable can hold its memory.
- An automatic lifetime variable or global variable activates when they are called else they vanish when the function executes.


Properties of variables
Scope
Lifetime
Location in the program where a variable is accessible
Location in the program where a variable exists

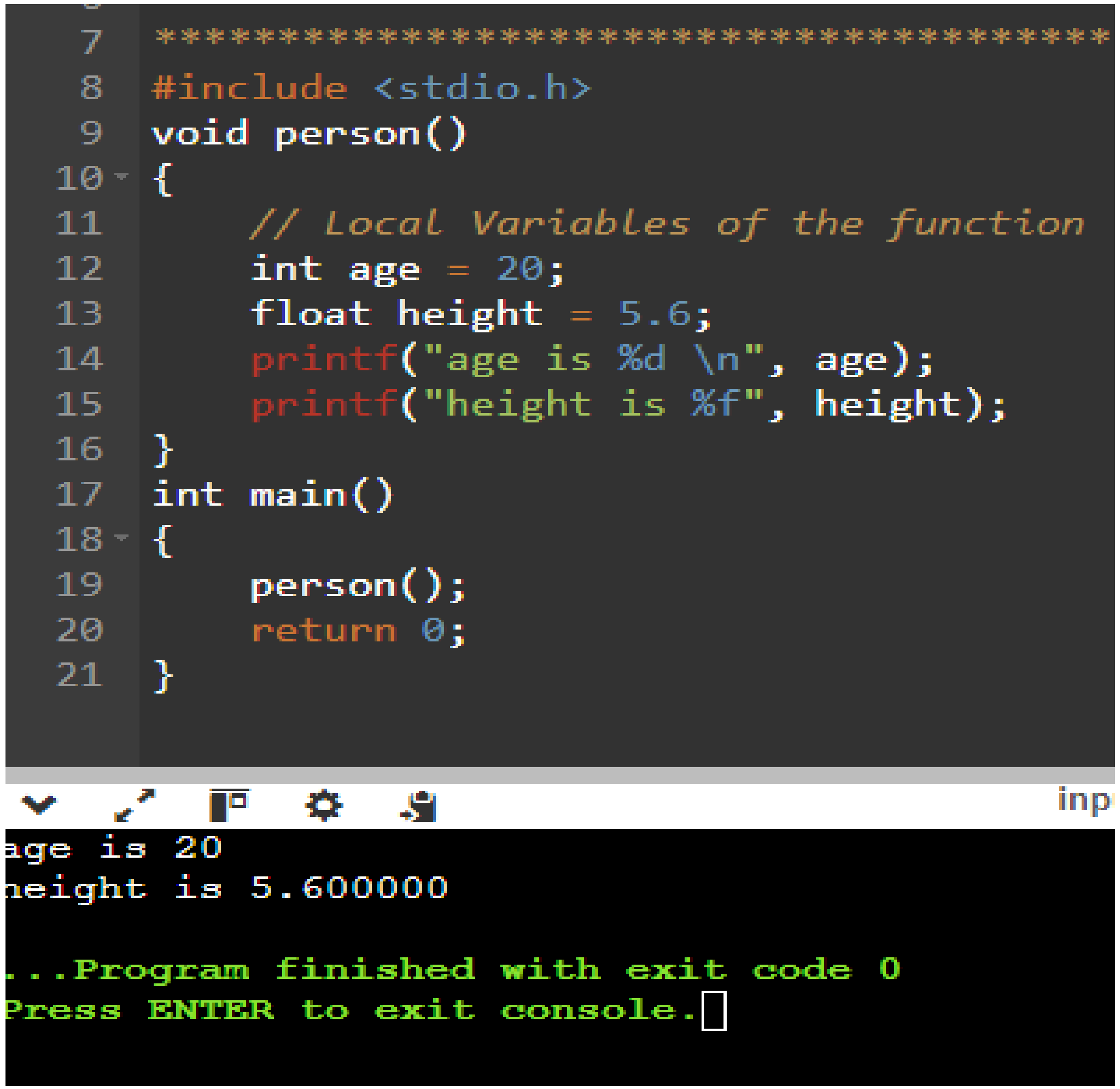
Types of Variables and its Scope

There are 5 types of variables in C language, which are:

1. **Local Variable**
2. **Global Variable**
3. **Static Variable**
4. **Automatic Variable**
5. **External Variable**

Local Variables

- Local variables are declared and initialized at **the start of a function** or block and allocated memory inside that execution scope.
- **The statements only inside that function can access that local variable.**
- Such variables get destroyed when the control exits from the function.

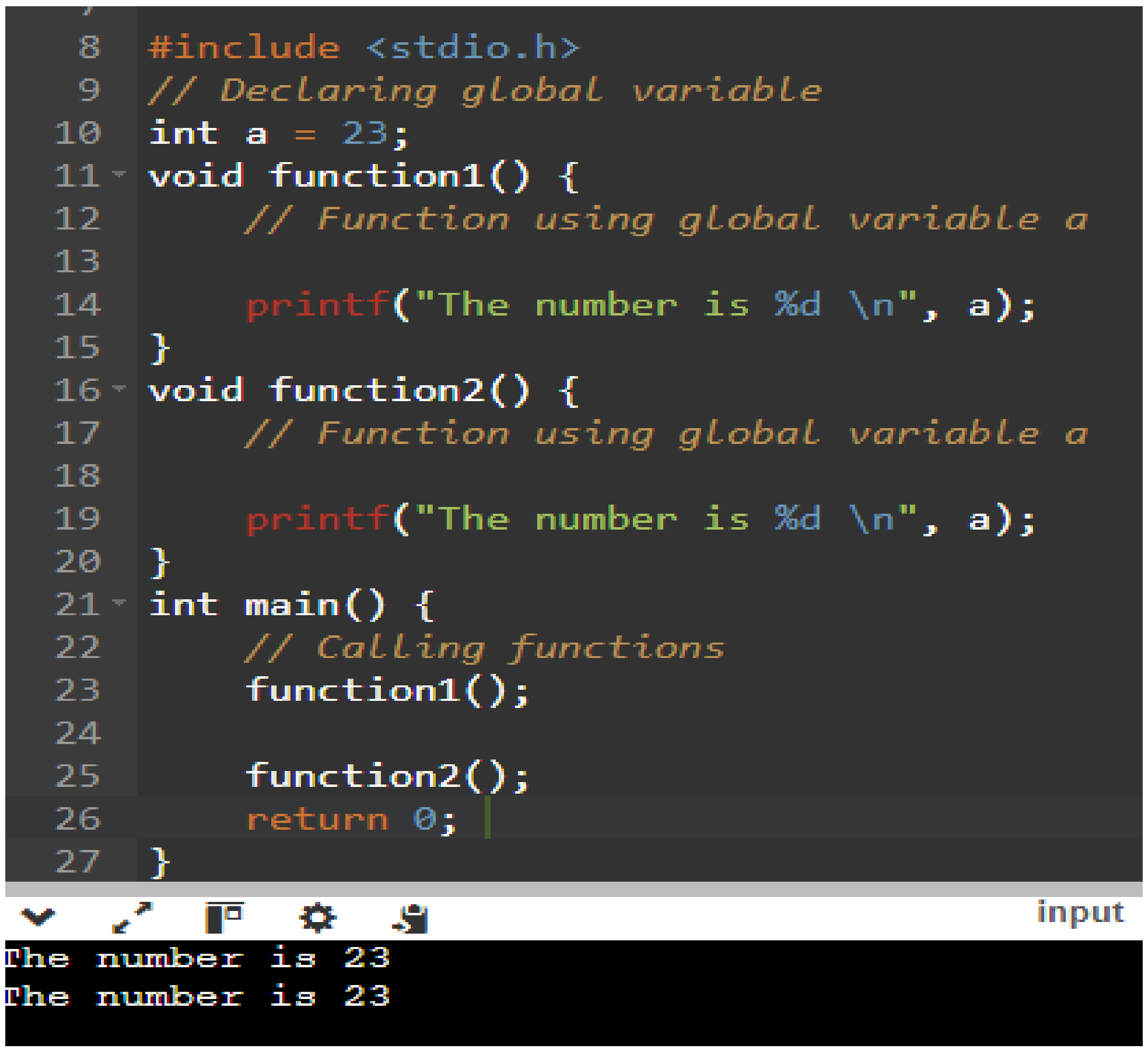


```c
*****************************************
#include <stdio.h>
void person()
{
    // Local Variables of the function
    int age = 20;
    float height = 5.6;
    printf("age is %d \n", age);
    printf("height is %f", height);
}
int main()
{
    person();
    return 0;
}
```

```
age is 20
neight is 5.600000

...Program finished with exit code 0
Press ENTER to exit console.
```

Global Variables

- The global variable in C is defined **outside of any function** and is therefore accessible to all functions.
- Global variables in C are **initialized when the program starts, and until the program exits**, these variables remain stored in the **memory.**

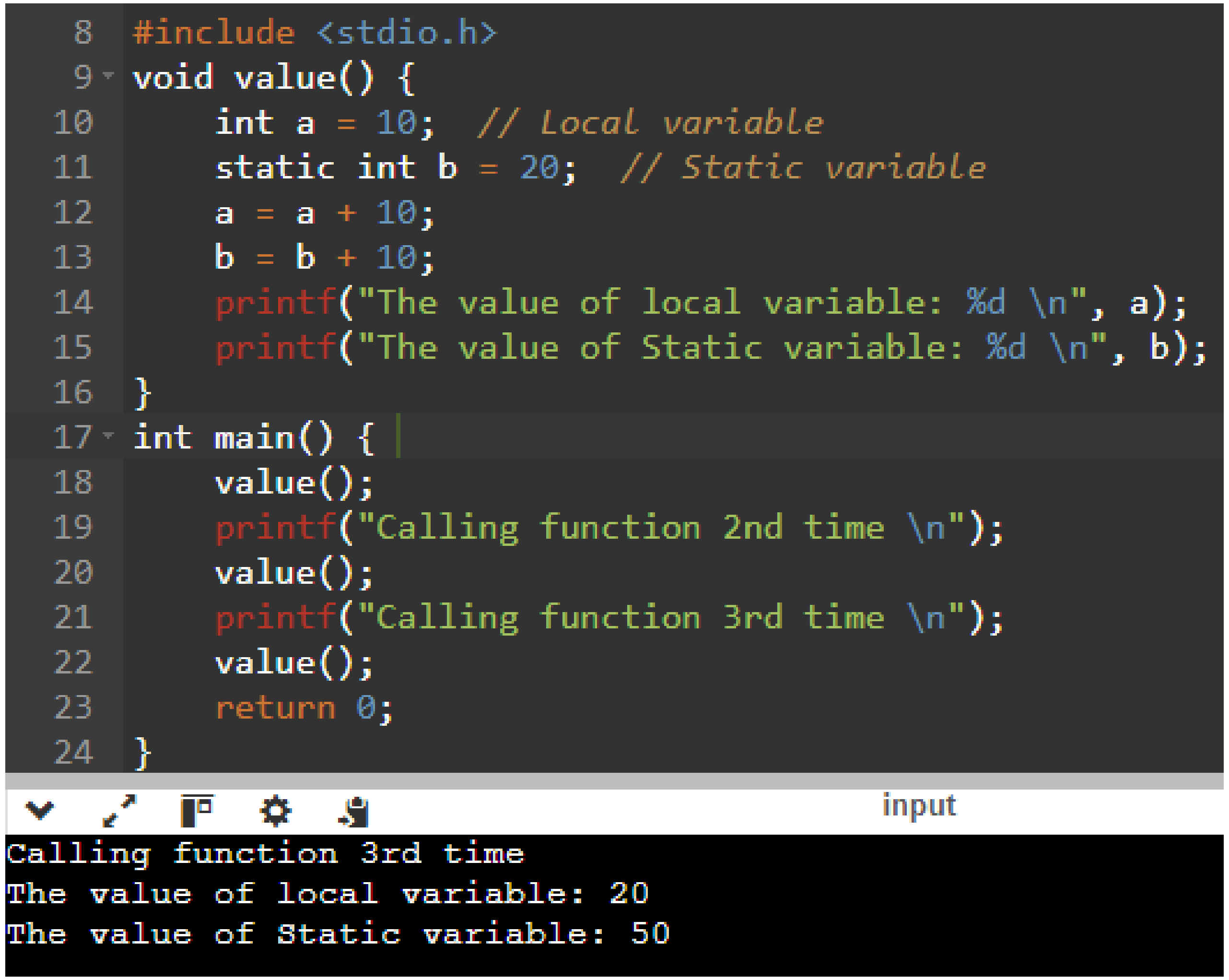


```c
#include <stdio.h>
// Declaring global variable
int a = 23;
void function1() {
     // Function using global variable a

     printf("The number is %d \n", a);
}
void function2() {
     // Function using global variable a

     printf("The number is %d \n", a);
}
int main() {
     // Calling functions
     function1();

     function2();
     return 0;
}
```

```
The number is 23
The number is 23
```

Static Variable in C

- Static variables enable data to remain after the function has been executed and returned; this allows the function to remember a value from a previous call, making it both efficient and powerful.
- We will learn in detail about static variables and functions in the section, Boolean and Static in C

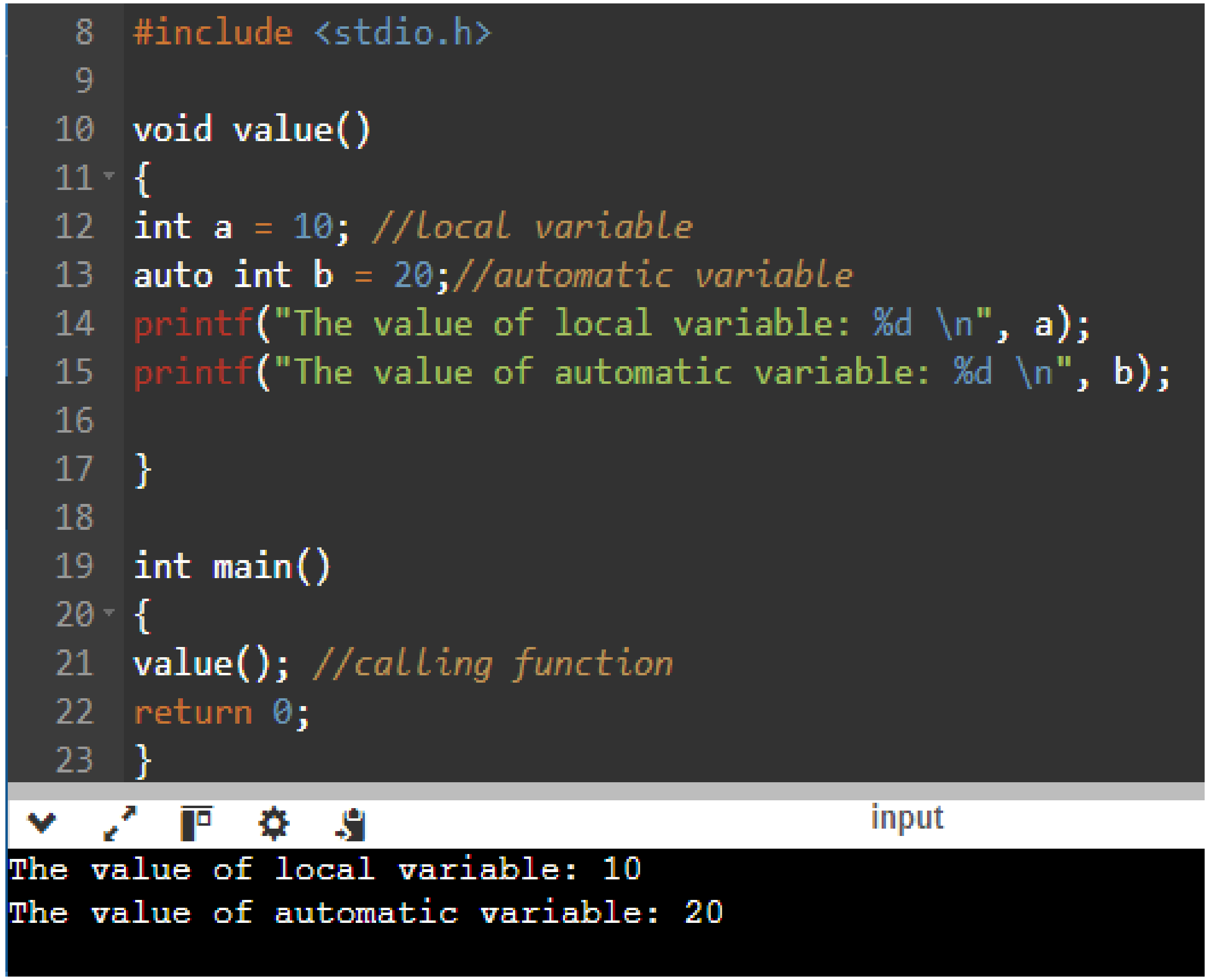


```c
#include <stdio.h>
void value() {
    int a = 10;  // Local variable
    static int b = 20;  // Static variable
    a = a + 10;
    b = b + 10;
    printf("The value of local variable: %d \n", a);
    printf("The value of Static variable: %d \n", b);
}
int main() {
    value();
    printf("Calling function 2nd time \n");
    value();
    printf("Calling function 3rd time \n");
    value();
    return 0;
}
```

```
Calling function 3rd time
The value of local variable: 20
The value of Static variable: 50
```

Automatic Variable in C

- All the local variables are automatic variables by default.
- They are also known as auto variables.
- All variables declared in C programming are automatic by default. You can use the auto keyword to declare the automatic variable.
- An automatic variable is a local variable whose lifetime is within the code only.

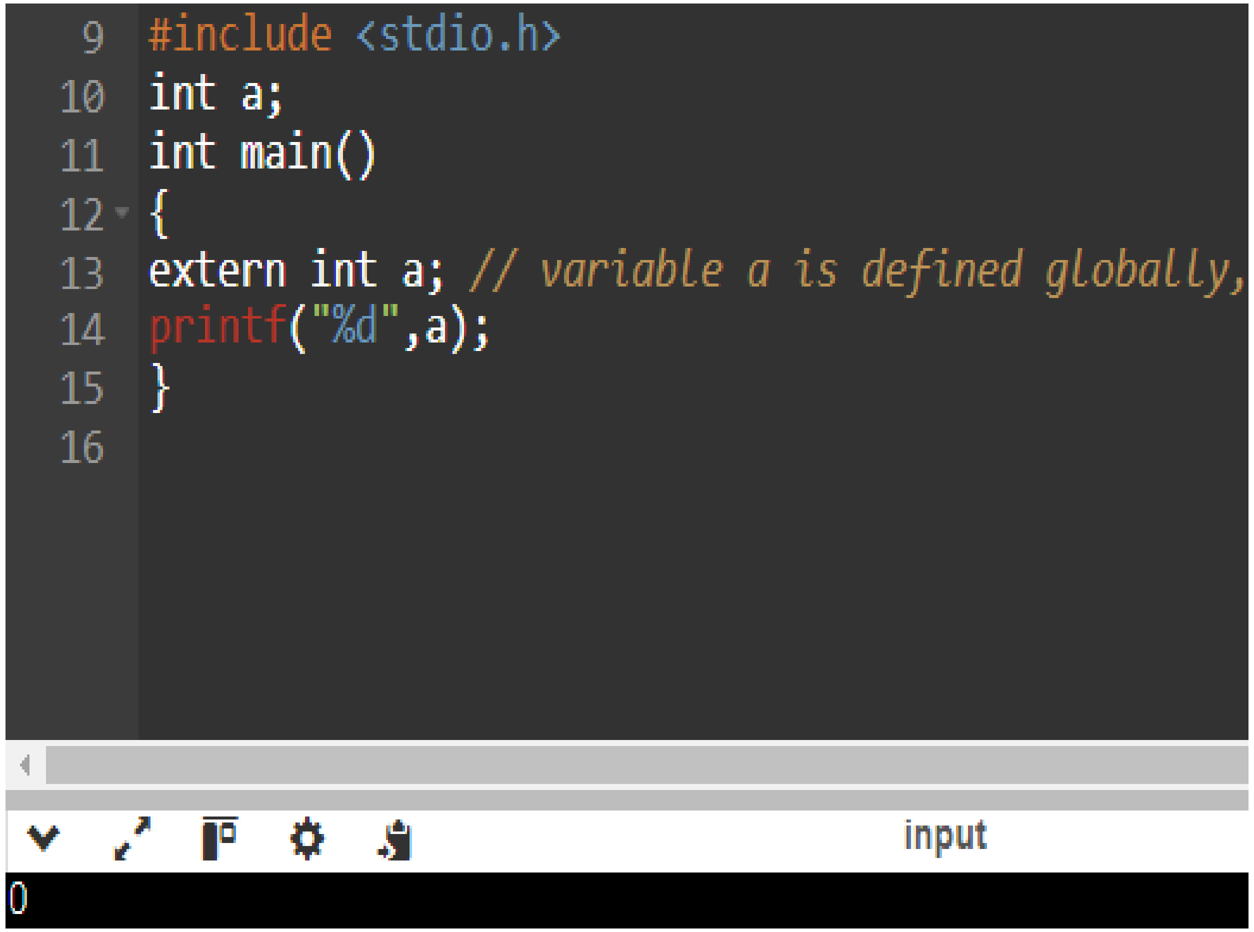


```c
#include <stdio.h>

void value()
{
int a = 10; //local variable
auto int b = 20;//automatic variable
printf("The value of local variable: %d \n", a);
printf("The value of automatic variable: %d \n", b);

}

int main()
{
value(); //calling function
return 0;
}
```

```
The value of local variable: 10
The value of automatic variable: 20
```

External Variable in C

- The extern variables use the extern keyword before the variable definition.
- This enhances the variable visibility, and the variable can be used in different C files.
- It is a global variable.
- It is not necessary to initialize the variable at the time of its declaration.
- Its lifetime is the entire program.

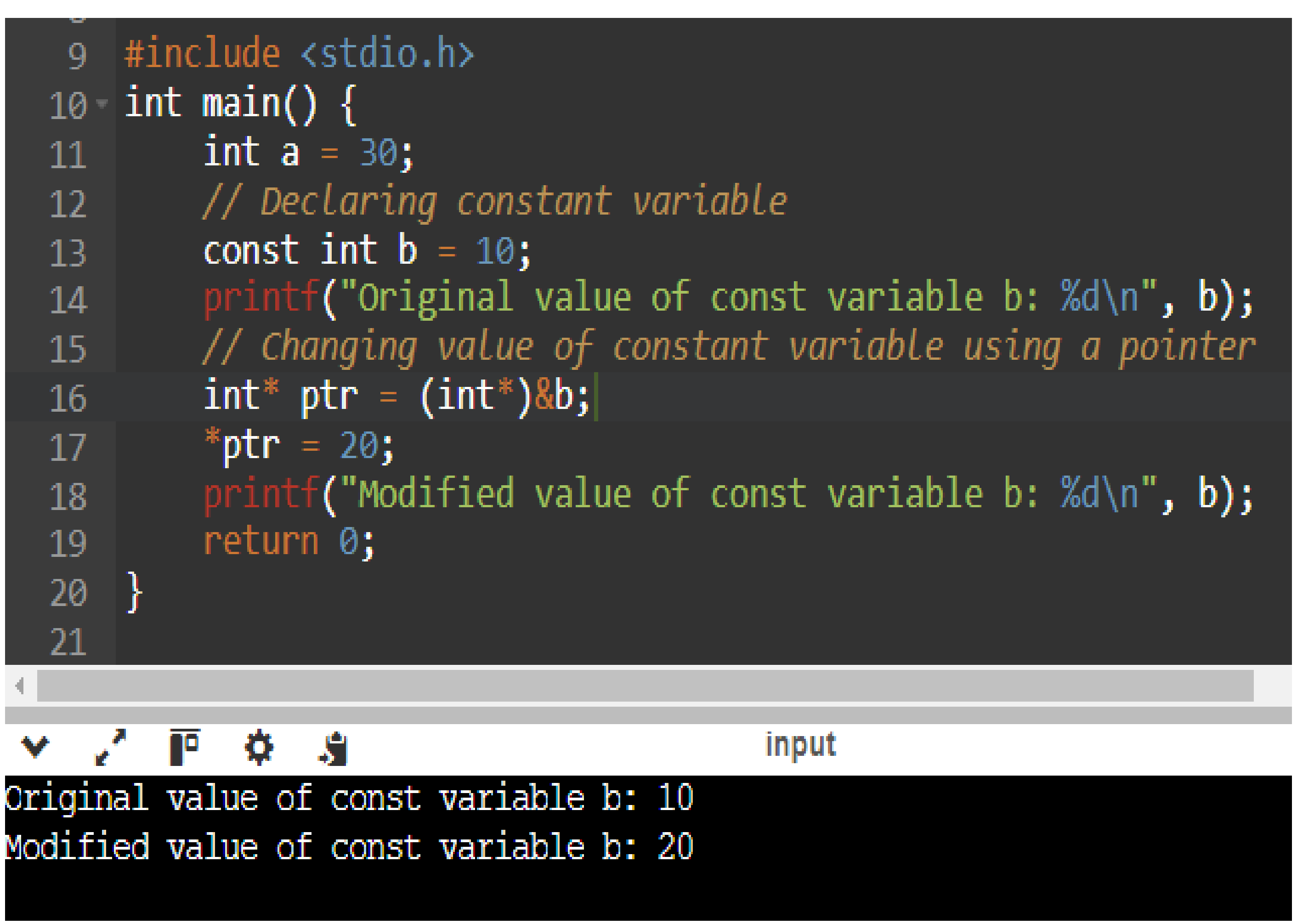


```c
#include <stdio.h>
int a;
int main()
{
extern int a; // variable a is defined globally,
printf("%d",a);
}
```

```
0
```

Constant Variables in C

- The constant variables are special variables in C whose value does not change throughout the program after initialization.
- It is a read-only variable in C.
- Initializing a constant variable at the time of variable declaration is mandatory.
- It uses the “const” keyword before its definition.



```c
#include <stdio.h>
int main() {
    int a = 30;
    // Declaring constant variable
    const int b = 10;
    printf("Original value of const variable b: %d\n", b);
    // Changing value of constant variable using a pointer
    int* ptr = (int*)&b;
    *ptr = 20;
    printf("Modified value of const variable b: %d\n", b);
    return 0;
}
```

```
Original value of const variable b: 10
Modified value of const variable b: 20
```

Reading Materials

LABs Link:

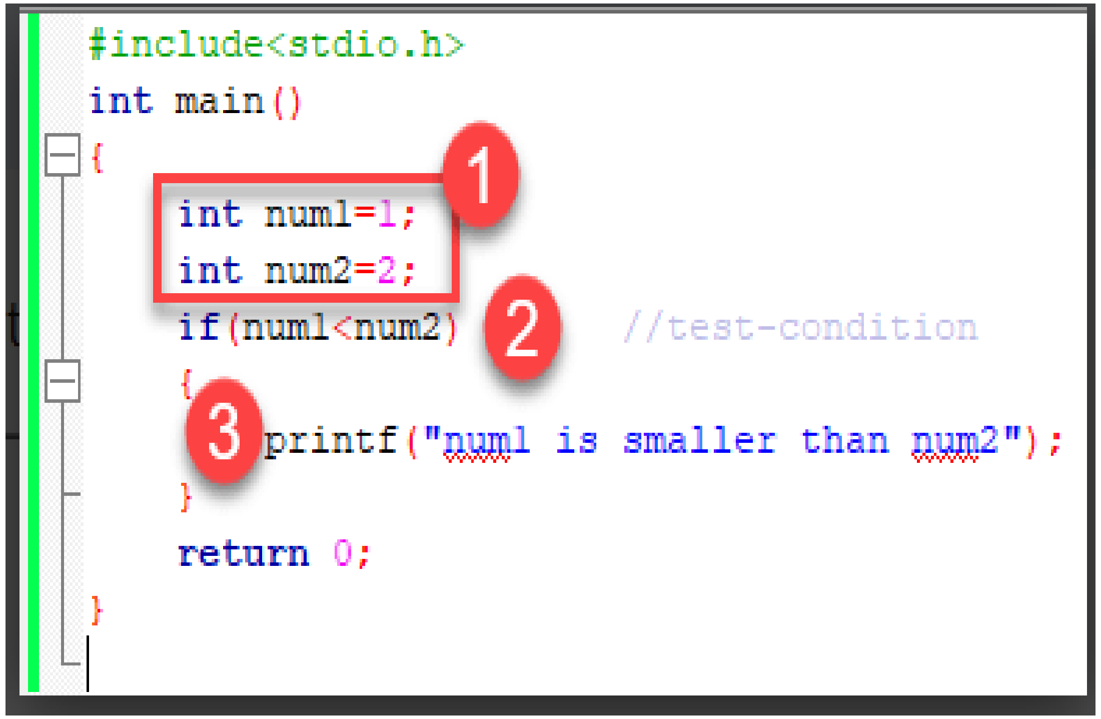
- https://www.scholarhat.com/tutorial/c/variables-in-c-language

Video Link:

- https://www.youtube.com/watch?v=Es0OziaqaOY

Statements in C

- A **computer program** is a list of "instructions" to be "executed" by a computer.
- In a programming language, these programming instructions are called **statements**.
- The following statement "instructs" the compiler to print the text "Hello World" to the screen:
- It is important that you end the statement with a semicolon ;

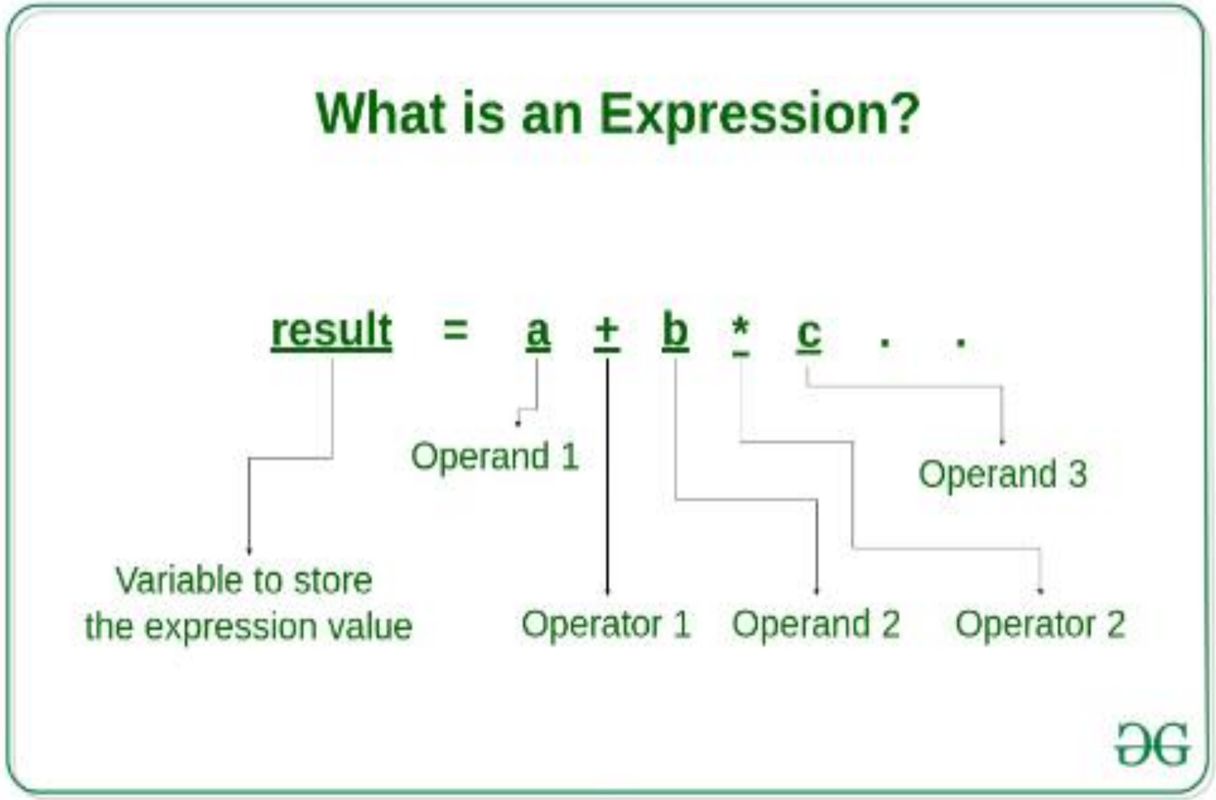


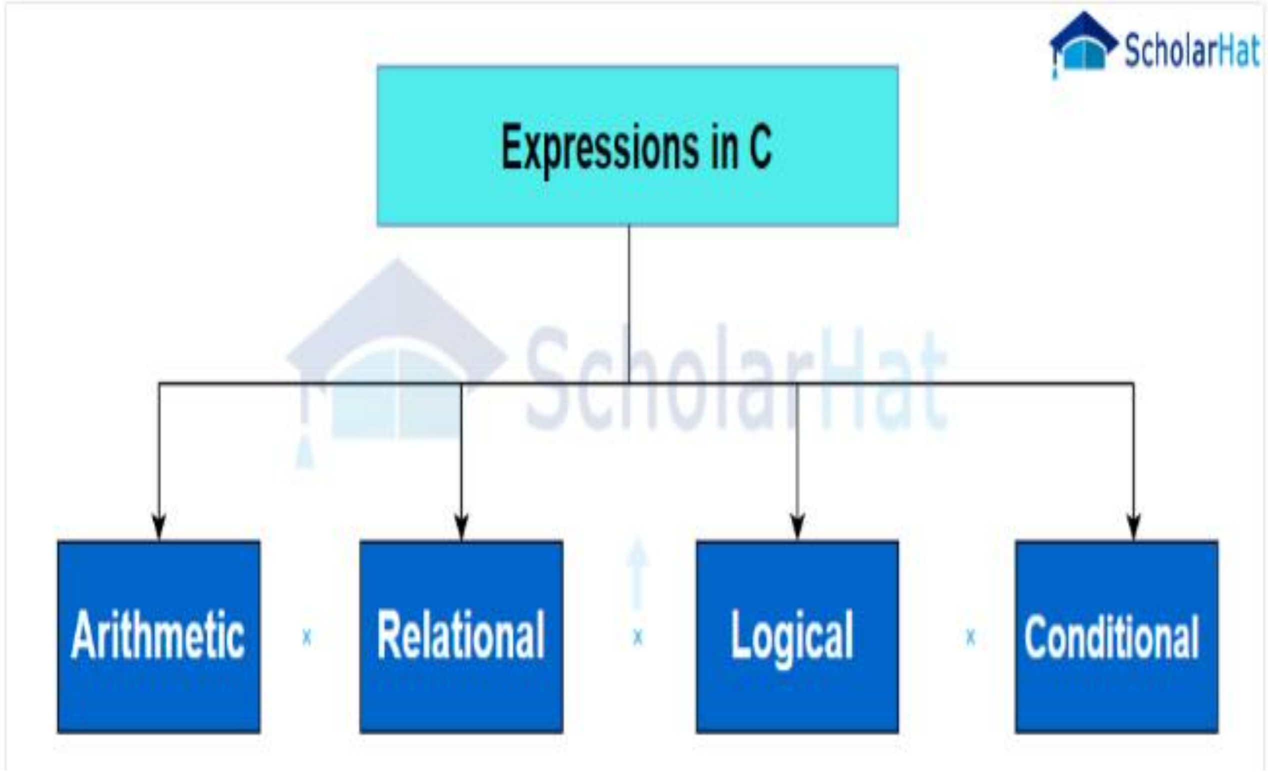
```
#include<stdio.h>
int main()
{
    int num1=1;
    int num2=2;
    if(num1<num2)              //test-condition
    {
        printf("num1 is smaller than num2");
    }
    return 0;
}
```

Expression in C

- An expression in C language is a combination of **symbols, numbers, and/or text** that produces a particular result.
- These expressions are evaluated and can be used to assign values to variables, perform mathematical operations, or execute different actions such as comparison and Boolean logic.
- Expressions in C consist of C language constants (numbers or strings), C language operators (arithmetic, assignment, relational), and C language variables representing values stored in computer memory.

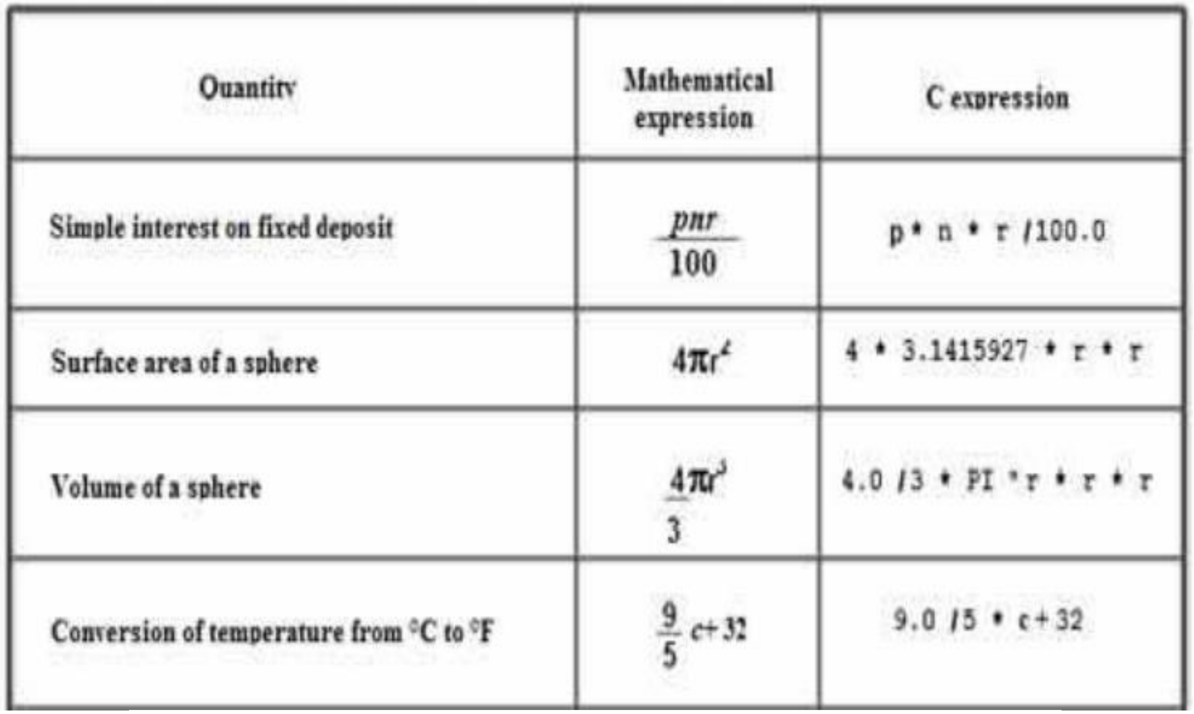
Types of Expressions in C





Arithmetic Expression in C

- An arithmetic expression consists of operands and arithmetic operators.
- It performs computations on the **int, float,** or **double** type values.
- Arithmetic operations can be performed in a single line of code or multiple lines combined with arithmetic operations such as **addition, subtraction, multiplication, and division**.

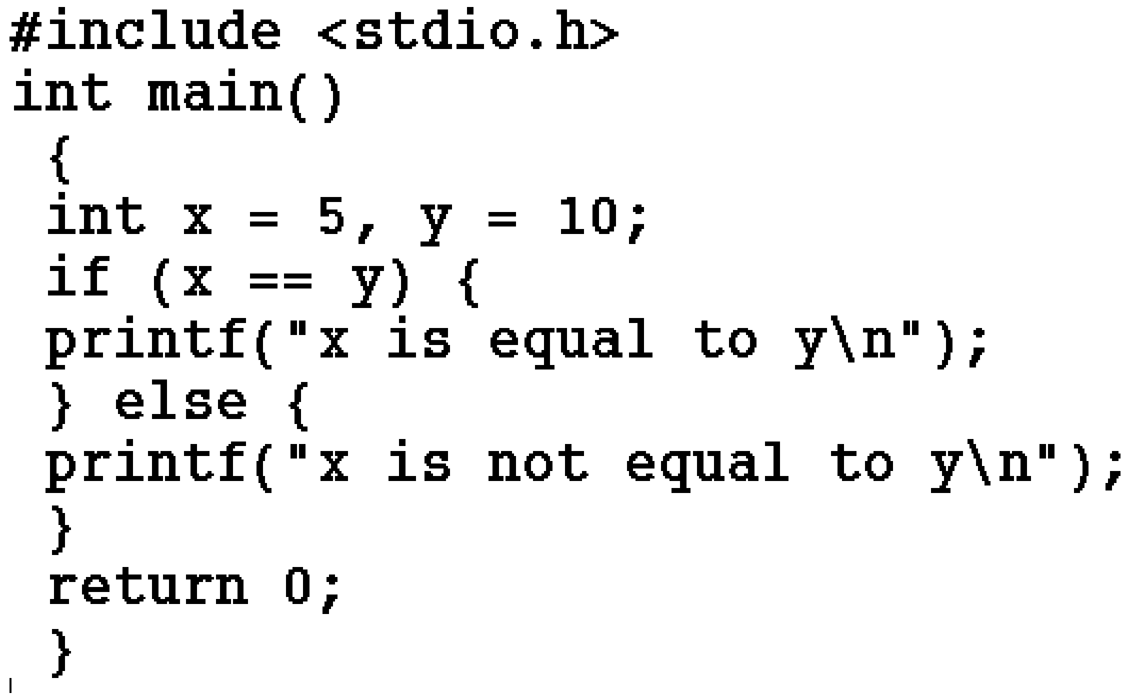


Quantity	Mathematical expression	C expression
Simple interest on fixed deposit	$\frac{pnr}{100}$	p * n * r /100.0
Surface area of a sphere	$4\pi r^2$	4 * 3.1415927 * r * r
Volume of a sphere	$\frac{4\pi r^3}{3}$	4.0 /3 * PI * r * r * r
Conversion of temperature from °C to °F	$\frac{9}{5}c+32$	9.0 /5 * c + 32

```
Arithmetic Expression:

a=2,b=3,c=4

Z = a + b - (a * c)
```

Relational Expression in C

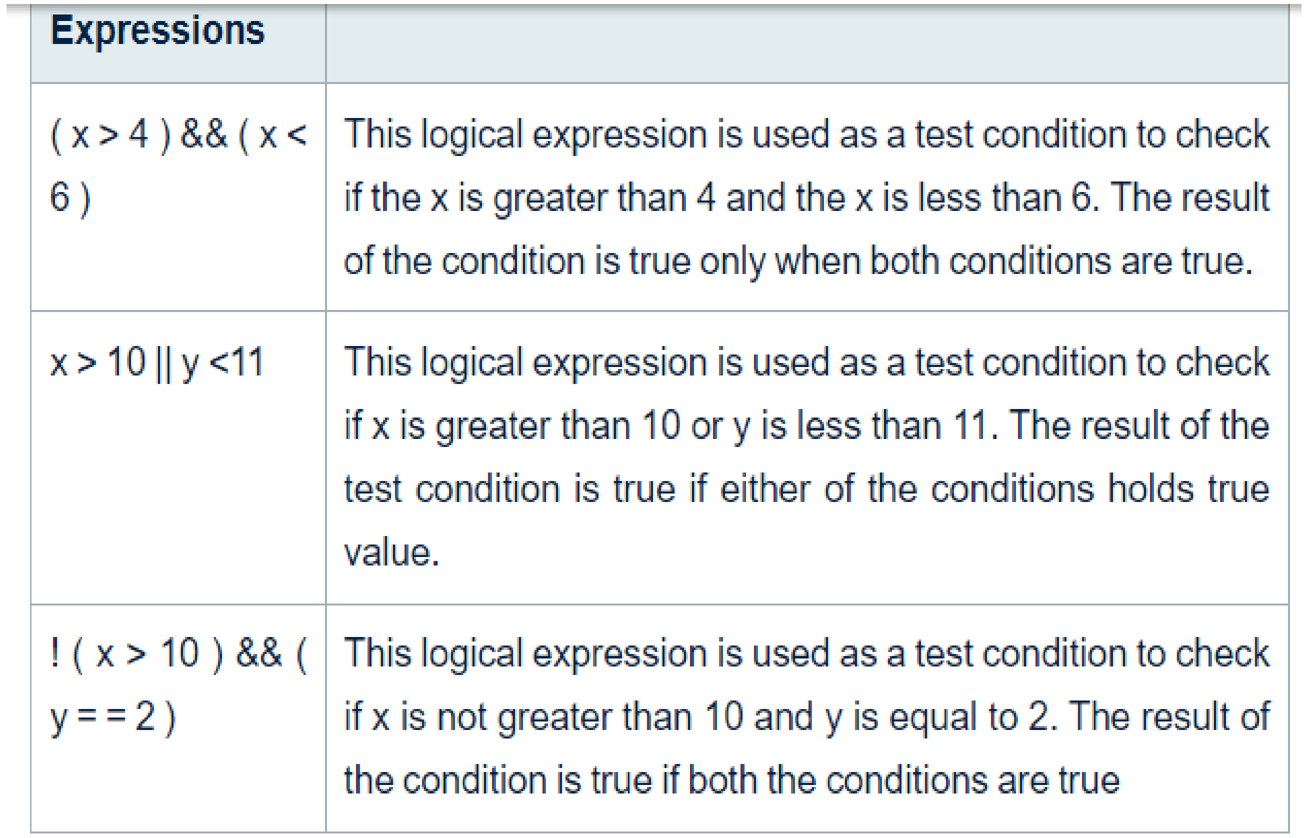
- Relational expressions use comparison operators such as **'>' (greater than) and '<' (less than)** to compare two operands.
- The result of the comparison is a boolean value i.e. 0(false) or non-zero (true)



```
#include <stdio.h>
int main()
 {
 int x = 5, y = 10;
 if (x == y) {
 printf("x is equal to y\n");
 } else {
 printf("x is not equal to y\n");
 }
 return 0;
 }
```

Logical Expression in C

- Logical expressions in C are a powerful tool for controlling the logic of the flow of the program.
- They are made by combining as many relational expressions as the programmer wants.
- It then determines if certain statements or groups of statements should be executed or not.



Expressions	
(x > 4) && (x < 6)	This logical expression is used as a test condition to check if the x is greater than 4 and the x is less than 6. The result of the condition is true only when both conditions are true.
x > 10 || y <11	This logical expression is used as a test condition to check if x is greater than 10 or y is less than 11. The result of the test condition is true if either of the conditions holds true value.
! (x > 10) && (y = = 2)	This logical expression is used as a test condition to check if x is not greater than 10 and y is equal to 2. The result of the condition is true if both the conditions are true

Logical Expression in C

```
#include <stdio.h>
int main()
{
 int x = 10;
 int y = 2;
 if ( (x >10) || (y<5))
 {
 printf("Condition is true");
 }
 else
 printf("Condition is false");
 return 0;
}
```

Conditional Expression in C

- The conditional expression consists of three operands.
- It returns **1** if the condition is true otherwise **0.**

```
#include <stdio.h>
#include <string.h>
int main()
{
 int age = 29;
 char status;
 status = (age>22) ? 'M': 'U';
 if(status == 'M')
 printf("Married");
 else
 printf("Unmarried");
 return 0; }
```