C++

Video 1:

Introduction to C++, Installing VS Code, g++ & more | C++ Tutorials for Beginners #1

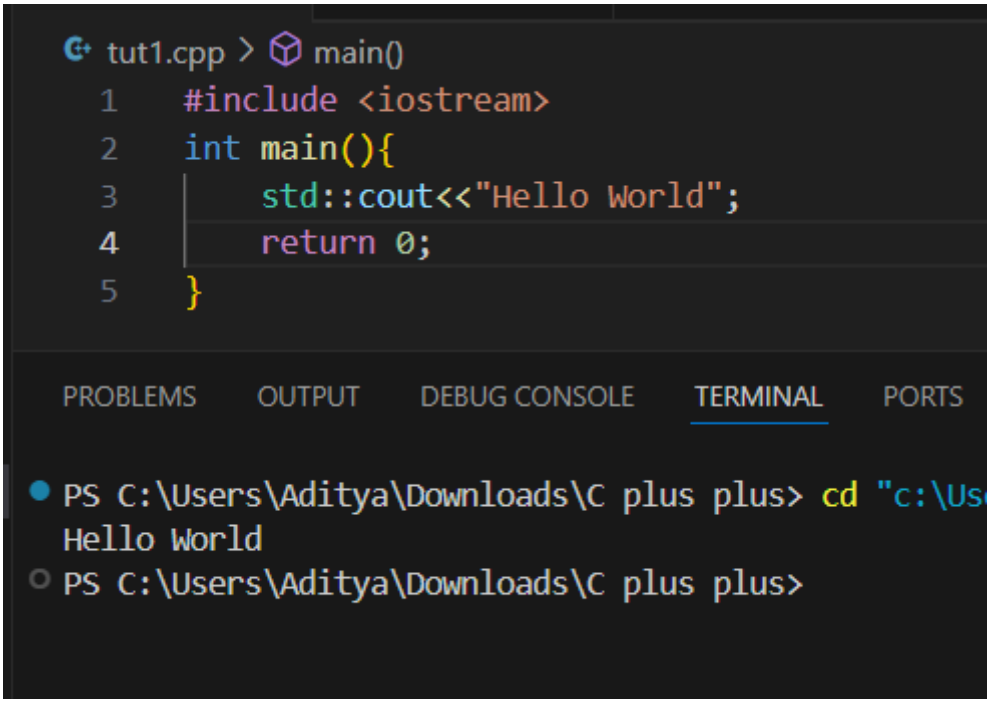
What is Programming?

Programming is the process of designing, writing, testing, and maintaining code to create software applications. It involves using programming languages to give instructions to a computer, enabling it to perform specific tasks. Programming requires logic, problem-solving, and creativity to develop efficient and functional solutions for various computational problems.

Why to use C++?

C++ is used because it offers a balance of performance, efficiency, and flexibility. It provides **high-speed execution**, **low-level memory control**, and **object-oriented programming (OOP)** features. C++ is widely used in **system programming, game development, embedded systems, real-time applications, and high-performance computing** due to its powerful standard library and portability.

First C++ program:



Video 2:

Basic Structure of a C++ Program | C++ Tutorials for Beginners #2

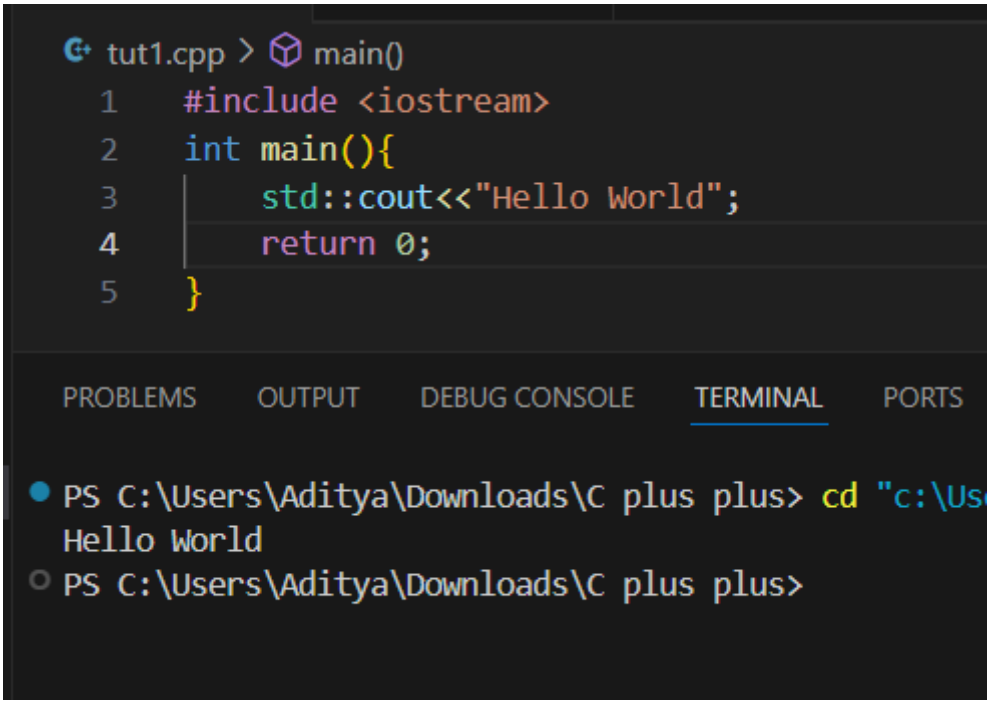
In which year major changes were made in C++?

C++ has undergone several major changes over the years. The most significant updates were:

1. **C++98 (1998)** – The first standardized version, introducing the STL (Standard Template Library).
2. **C++03 (2003)** – Minor bug fixes and performance improvements.
3. **C++11 (2011)** – A major update introducing smart pointers, move semantics, auto keyword, lambda expressions, and multithreading.
4. **C++14 (2014)** – Improved usability with small enhancements like generalized lambda expressions and relaxed constexpr.
5. **C++17 (2017)** – Added structured bindings, filesystem library, and performance optimizations.
6. **C++20 (2020)** – Introduced concepts, coroutines, ranges, modules, and a new standard library.
7. **C++23 (2023)** – Further refinements, adding improvements like explicit object parameters and better compile-time features.

Each update brought new features to make C++ more powerful, efficient, and easier to use.

A basic C++ program:



1. Preprocessor Directive:

#include <iostream>

- #include <iostream> allows the use of **input-output operations** (like std::cout for printing).

- The **preprocessor** includes the iostream library before compilation.

2. Main Function:

int main() {

- main() is the **entry point** of every C++ program.
- The program execution starts from the main() function.

3. Output Statement:

std::cout << "Hello World";

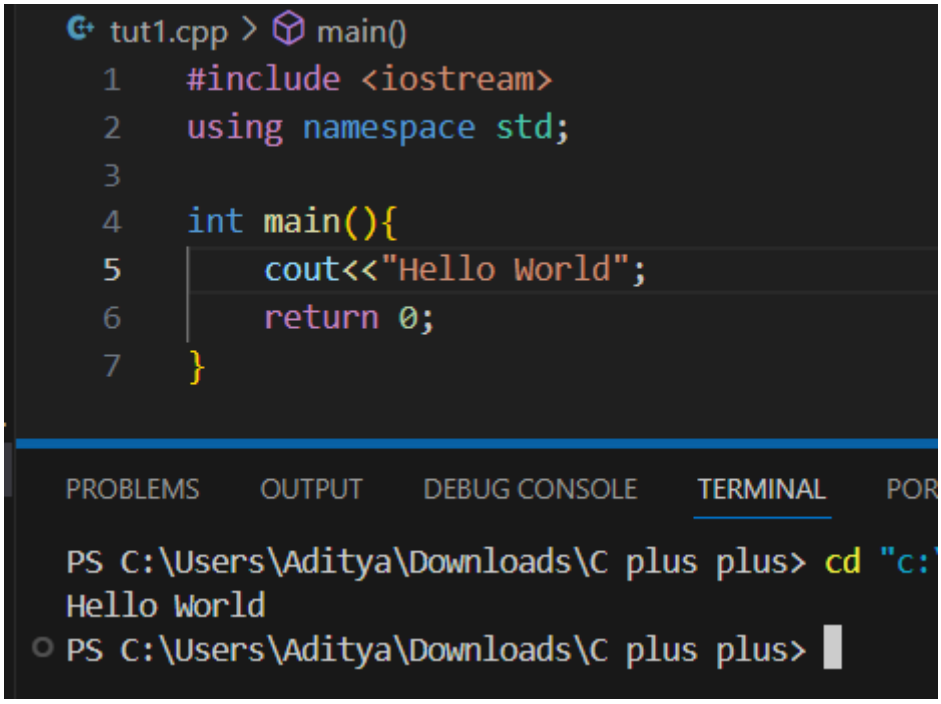
- std::cout (console output) prints text to the screen.
- << is the **insertion operator** used to pass data to std::cout.
- "Hello World" is a string literal that gets printed.

4. Return Statement:

return 0;

- return 0; indicates **successful program execution**.
- In int main(), returning 0 signals the operating system that the program ran successfully.

Above code can also be written as:



Video 3:

Variables & Comments in C++ in Hindi | C++ Tutorials for Beginners #3

What is the basic difference between low-level and high-level language?

Feature	Low-Level Language	High-Level Language
Abstraction	Close to hardware, minimal abstraction	More abstract, closer to human language
Ease of Use	Difficult to learn and write	Easier to learn and write
Performance	Faster execution, optimized for hardware	Slower due to abstraction and compilation
Portability	Hardware-dependent (not portable)	Platform-independent (portable)
Examples	Assembly, Machine Code	C, C++, Python, Java

What is a variable in C++?

A **variable** in C++ is a named storage location in memory that holds a value, which can change during program execution. It has a **data type** (e.g., int, float, char) that defines the kind of data it stores. Variables enable dynamic data manipulation and are fundamental to programming logic.

Types of data type in C++:

In C++, data types are classified into the following categories:

1. Primary (Built-in) Data Types:

Type	Description	Example
int	Stores integers	int x = 10;
float	Stores decimal numbers (single precision)	float y = 3.14;
double	Stores large decimal numbers (double precision)	double z = 9.8765;
char	Stores single characters	char c = 'A';
bool	Stores Boolean values (true or false)	bool flag = true;

2. Derived Data Types:

Type	Description	Example
array	Collection of elements of the same type	int arr[5] = {1, 2, 3, 4, 5};
pointer	Stores memory addresses	int* ptr = &x;
reference	Alias for another variable	int &ref = x;

3. User-Defined Data Types:

Type	Description	Example
struct	Groups related variables	struct Student { int age; };
class	Defines objects with attributes & methods	class Car { public: string model; };
union	Stores different data types in the same memory	union Data { int x; float y; };
enum	Defines named constant values	enum Color { RED, GREEN, BLUE };

4. Abstract Data Types:

Type	Description	Example
function	Encapsulates reusable code	void greet() { cout << "Hello"; }

Basic example of using variable in C++::

```
tut3.cpp > main()
#include <iostream>
using namespace std;

int main(){
    int sum = 6;
    cout << " The value stored in variable sum is " << sum;
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plus
 plus\" ; if ($?) { g++ tut3.cpp -o tut3 } ; if ($?) { .\tut3 }
 The value stored in variable sum is 6
PS C:\Users\Aditya\Downloads\C plus plus>
```

What is comment in C++?

Comments in C++ are **non-executable** text used to explain code, making it easier to understand. The compiler ignores comments during execution.

A basic example of single-line comment: Starts with //

```
tut3.cpp > main()

int main(){
    // Hey I am a comment
    cout << "Hey Aditya";
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Adit
 plus\" ; if ($?) { g++ tut3.cpp -o tut3 } ; if ($?) { .\tu
Hey Aditya
PS C:\Users\Aditya\Downloads\C plus plus>
```

A basic example of Multi-line Comment: Enclosed between /* ... */

```
int main(){
    /*
    Hey this is a multi line comment
    You can use it instead of using the single line comment multiple times
    */
    cout << "Hey Aditya";
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plus
 plus\" ; if ($?) { g++ tut3.cpp -o tut3 } ; if ($?) { .\tut3 }
Hey Aditya
PS C:\Users\Aditya\Downloads\C plus plus>
```

Code
Code
Code
Code
Code

Video 4:

Variable Scope & Data Types in C++ in Hindi | C++ Tutorials for Beginners #4

What is a variable?

A variable in C++ is a named storage location in memory that holds a value, which can be modified during program execution. It has a specific data type (e.g., int, float, char) that defines the kind of data it stores, enabling dynamic data manipulation in a program.

Syntax:

data_type variable_name = value; // Optional initialization

data_type variable1 = value1, variable2 = value2; //two variable in one line

A basic example:

```
tut4.cpp > main()
#include <iostream>
using namespace std;

int main(){
    int a = 3;
    int b = 4, c = 9;
    cout << "a: " << a << ",c: " << c;
    return 0;
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C pl
 plus\" ; if ($?) { g++ tut4.cpp -o tut4 } ; if ($?) { .\tut4 }
a: 3,c: 9
PS C:\Users\Aditya\Downloads\C plus plus>
```

Difference Between Types of Variables Based on Scope in C++

Variable Type	Scope	Usage	Example
Local Variable	Inside a function or block	Accessible only within that function/block	void func() { int x = 10; }
Global Variable	Outside all functions	Accessible throughout the program	int x = 10; int main() { cout << x; }

Variable Type	Scope	Usage	Example
Static Variable	Inside a function, retains value between calls	Preserves its value between function calls	void func() { static int x = 0; x++; }
Register Variable	Stored in CPU register for fast access	Used for frequently accessed variables	register int count = 0;
Extern Variable	Declared in one file, defined in another	Used for global access across multiple files	extern int x;

Local variable:

A local variable is declared inside a function or block and can only be accessed within that function or block. It cannot be used outside its declared scope.

Example a:

```
//local variable
int main(){
    int a = 5;
    cout << "a is: "<< a;
    return 0;
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plu
 plus\" ; if ($?) { g++ tut4.cpp -o tut4 } ; if ($?) { .\tut4 }
a is: 5
PS C:\Users\Aditya\Downloads\C plus plus>
```

Example b:

```cpp
//local varibale exmaple 2
void display() {
    int num = 10;  // Local variable (only accessible inside display())
    cout << "Local variable: " << num << endl;
}

int main() {
    display();
    // cout << num;  // ❌ Error: 'num' is not accessible here
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plus
 plus\" ; if ($?) { g++ tut4.cpp -o tut4 } ; if ($?) { .\tut4 }
Local variable: 10
PS C:\Users\Aditya\Downloads\C plus plus>
```

Example c:

```cpp
int main() {
    {  // Start of a block
        int y = 20;  // Local variable inside the block
        cout << "Value of y: " << y << endl;
    }  // End of block

    // cout << y;  ❌ ERROR: 'y' is not accessible outside the block

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C
 plus plus\" ; if ($?) { g++ tut4.cpp -o tut4 } ; if ($?) { .\tut4 }
Value of y: 20
PS C:\Users\Aditya\Downloads\C plus plus>
```

Example d:

```
int main(){
    for (int i = 0; i < 3; i++)
    { //'i' is local to the loop
        cout << "i is: " << i << endl;
    }
    //cout << i; // 'i' can't be accessed outside the loop
    return 0;

}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C
 plus plus\" ; if ($?) { g++ tut4.cpp -o tut4 } ; if ($?) { .\tut4 }
i is: 0
i is: 1
i is: 2
PS C:\Users\Aditya\Downloads\C plus plus>
```

Example e:

```
int main(){
    int count = 0;
    while (count < 3)
    {
        int temp = count*2; //local variable is temp
        cout << " Temp value is: " << temp << endl;
        count++;
    }
    //cout << temp;
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plus
 plus\" ; if ($?) { g++ tut4.cpp -o tut4 } ; if ($?) { .\tut4 }
 Temp value is: 0
 Temp value is: 2
 Temp value is: 4
PS C:\Users\Aditya\Downloads\C plus plus>
```

Example f:

```
tut4.cpp > main()
int main(){
    int choice = 2;
    switch (choice)
    {
    case 1:{
        int a = 10; //local variable inside case 1
        cout << "a is:" <<a;
        break;
    }
    case 2:{
        int b = 2;
        cout << "b is:" <<b;
        break;
    }
    default:
        break;
    }
    //cout << b;
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plus plus\" ; if ($?) { g++ tut4.cpp -o tut
b is:2
PS C:\Users\Aditya\Downloads\C plus plus>
```

Example g:

```
int main (){
    int var=2;
    if (var>1)
    {
        int z = 3; //variable local to if
        cout << "The variable is " << z;
    }
    // cout<<z; //z is not accessible outside the 'if'
    return 0;
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plus plus\" ; if ($?) { g++ tut4.cpp -o tut4 }
The variable is 3
PS C:\Users\Aditya\Downloads\C plus plus>
```

Some examples of data type:

```cpp
// Data types
int main(){
    //integer varaible
    int a = 4, b = 5;
    cout << "a is:" << a << " b is:" <<b << endl;
    //float variable
    float c = 5.45453;
    cout << "c is: " << c <<"\n";
    //character variable
    char d = 'h';
    cout << d;
    return 0;

}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plus plus\" ; if ($?) {
a is:4 b is:5
c is: 5.45453
h
PS C:\Users\Aditya\Downloads\C plus plus>
```

Global variable:

Example a:

```cpp
//Global variable
#include <iostream>
using namespace std;

int globalVar = 10;  // Global variable

int main() {
    cout << "Global variable: " << globalVar << endl;
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plus plus\" ; if ($?) { g++ tut4.cpp -o tut4 }
Global variable: 10
PS C:\Users\Aditya\Downloads\C plus plus>
```

Example b:

```cpp
//modification of global variable
#include <iostream>
using namespace std;

int counter = 0;  // Global variable

void increment() {
    counter++;  // Modifying the global variable
}

void display() {
    cout << "Counter: " << counter << endl;
}

int main() {
    increment();
    display();
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plus plus\" ; if ($?) { g++ tut4.cpp -o tut4 } ; if ($?) { .\tut4 }
Counter: 1
PS C:\Users\Aditya\Downloads\C plus plus>
```

Rules for Declaring Variables in C++

1. **Must Start with a Letter or Underscore (_)**
    - The variable name **cannot** start with a number or special character.
2. **Can Contain Letters, Digits, and Underscore (_)**
    - Variables can include **letters (A-Z, a-z)**, **digits (0-9)**, and **underscores (_)**.
3. **Cannot Be a Reserved Keyword**
    - Words like int, float, return, etc., **cannot** be used as variable names.
4. **Case-Sensitive**
    - age, Age, and AGE are **different** variables.
5. **Must Be Declared Before Use**
    - A variable **must** be declared before it is used in the program.
6. **No Special Characters (Except _)**
    - Symbols like @, #, $, %, & are **not allowed** in variable names.
7. **Cannot Have Spaces**
    - Use camelCase (userName) or underscores (user_name) instead.
8. **Can Be Initialized at Declaration**
    - You **can** assign a value when declaring a variable.
9. **Should Have a Meaningful Name**
    - Use descriptive names like totalAmount instead of x.
10. **Scope and Lifetime Depend on Declaration Place**

- Variables declared inside functions are **local**, while those outside are **global**.

Video 5:

C++ Basic Input/Output & More | C++ Tutorials for Beginners #5

Ct+ comes with libraries which helps us in performing input/output. In C++ sequence of bytes corresponding to input and output are commonly known as streams.

Input Stream: Direction of flow of bytes takes place from input device (for ex Keyboard) to the main memory.

Output Stream: Direction of flow of bytes takes place from main memory to the output device (for example Display)

1. **Input (cin)**

   Used to take user input.

   The >> operator (extraction operator) is used to take input.

   2. Output (cout)

   Used to display output.

   The << operator (insertion operator) is used to print values.

   3. Including the iostream Library

   #include <iostream> must be included at the beginning.

   using namespace std; can be used to avoid writing std:: before cin and cout.

A basic example to show that:

```
int main(){
    /* What is an insertion operator? */
    /* << is called as insertion operator*/
    /* What is an extraction operator?*/
    /* >> is called as extratcion operator*/

    //program to take input and print that input
    int num1, num2;
    cout << "Enter the value of num1: \n";
    cin >> num1;

    cout << "Enter the value of num2: \n";
    cin >> num2;

    cout << "The sum is " << num1+num2;
    return 0;
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

 plus\" ; if ($?) { g++ tut5.cpp -o tut5 } ; if ($?) { .\tut5 }
Enter the value of num1:
55
Enter the value of num2:
33
The sum is 88
PS C:\Users\Aditya\Downloads\C plus plus>
```

1. **Using endl**

- endl moves to a new line (same as \n).

2. **Multiple Inputs in One Line**

- cin >> a >> b; takes multiple inputs separated by space.

3. **Taking String Input (getline())**

- Use getline(cin, variable); for multi-word input.

Example for multiple inputs in one line:

```cpp
#include <iostream>
using namespace std;

int main() {
    int age;
    float height;

    cout << "Enter your age and height (in cm): ";
    cin >> age >> height;  // Multiple inputs in one line

    cout << "You are " << age << " years old and " << height << " cm tall." << endl;

    return 0;
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Aditya\Downloads\C plus plus> cd "c:\Users\Aditya\Downloads\C plus plus\" ; if ($?) { g++ tut5.cpp -o tut5 } ; if ($?) { .\tut5
Enter your age and height (in cm): 120
44
You are 120 years old and 44 cm tall.
PS C:\Users\Aditya\Downloads\C plus plus>
```

What are reserved keywords?

Reserved keywords in C++ are predefined words with special meanings that cannot be used as variable names or identifiers. Examples include int, float, return, class, if, else, while, and for.

Data type and size in C++:

DATA TYPE	SIZE (IN BYTES)	RANGE
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
long long int	8	-(2^63) to (2^63)-1
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	
double	8	
long double	12	
wchar_t	2 or 4	1 wide character