

some  
fundamental  
of  
programing using c++

# Content

**1-Introduction**

**2.Variables and Data Types**

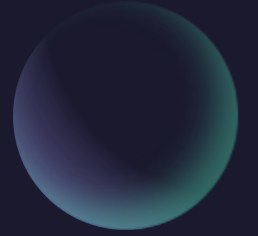
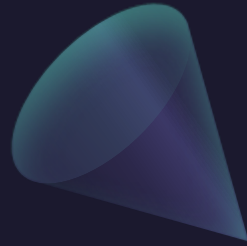
**3.Basic Operations**


**4. Input and Output**

**5. Conditional Statements**

**6. Loops**

**7. Functions**

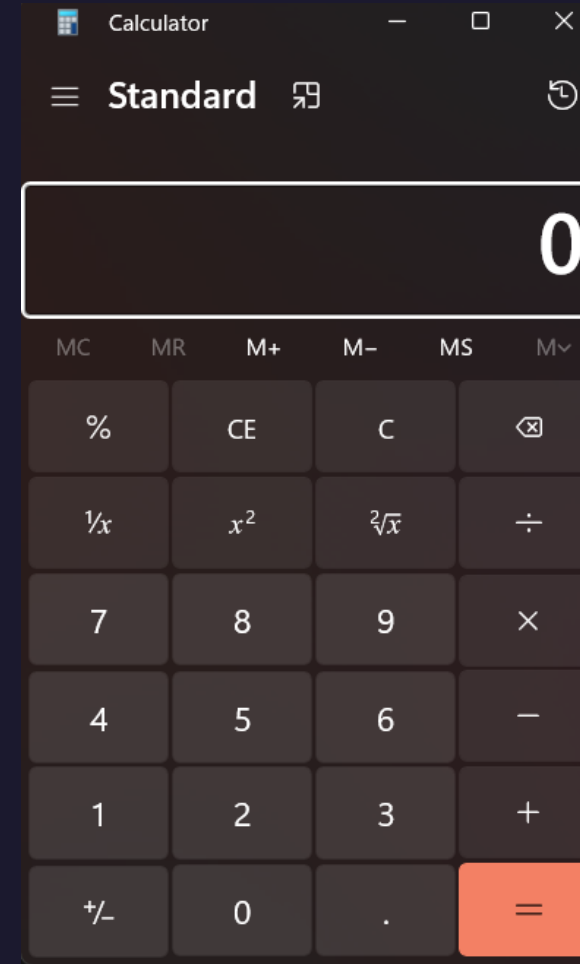




Programing language:  
a way to talk to the computer.

# Program is a set of Instructions

- Read number 1
- Read operations
- Read number 2
- Execute operation
- E.g  $3+5=8$



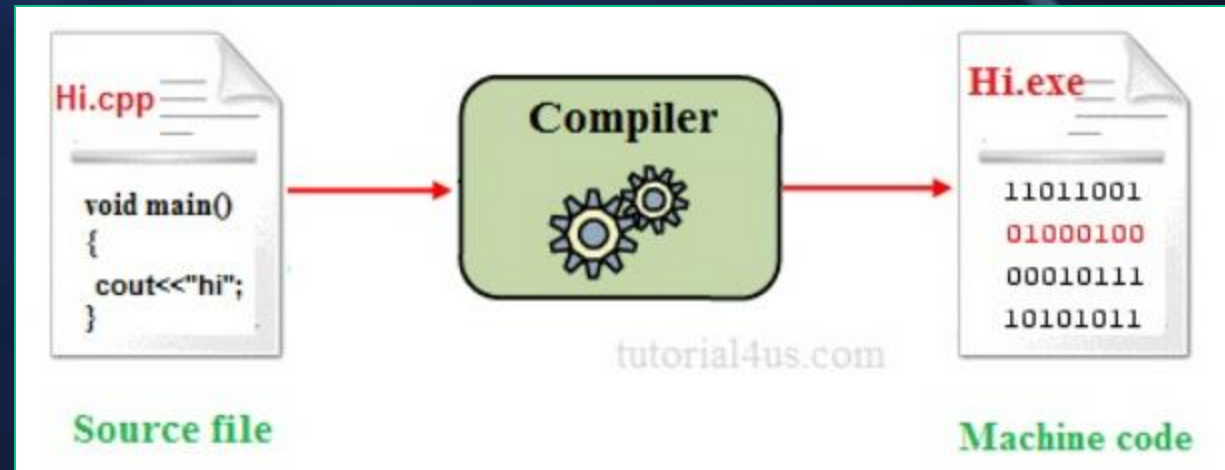
# Why Learn C++?

- **Efficiency:** C++ gives you fine-grained control over system resources like memory and CPU, allowing you to write high-performance applications.
- **Wide Application:** It is used in developing operating systems, game engines, desktop applications, and high-performance servers.
- **Object-Oriented Programming:** C++ supports OOP, which encourages organized, modular, and reusable code.
- **Standard Library:** The Standard Template Library (STL) in C++ provides a rich set of algorithms and data structures for handling complex operations.

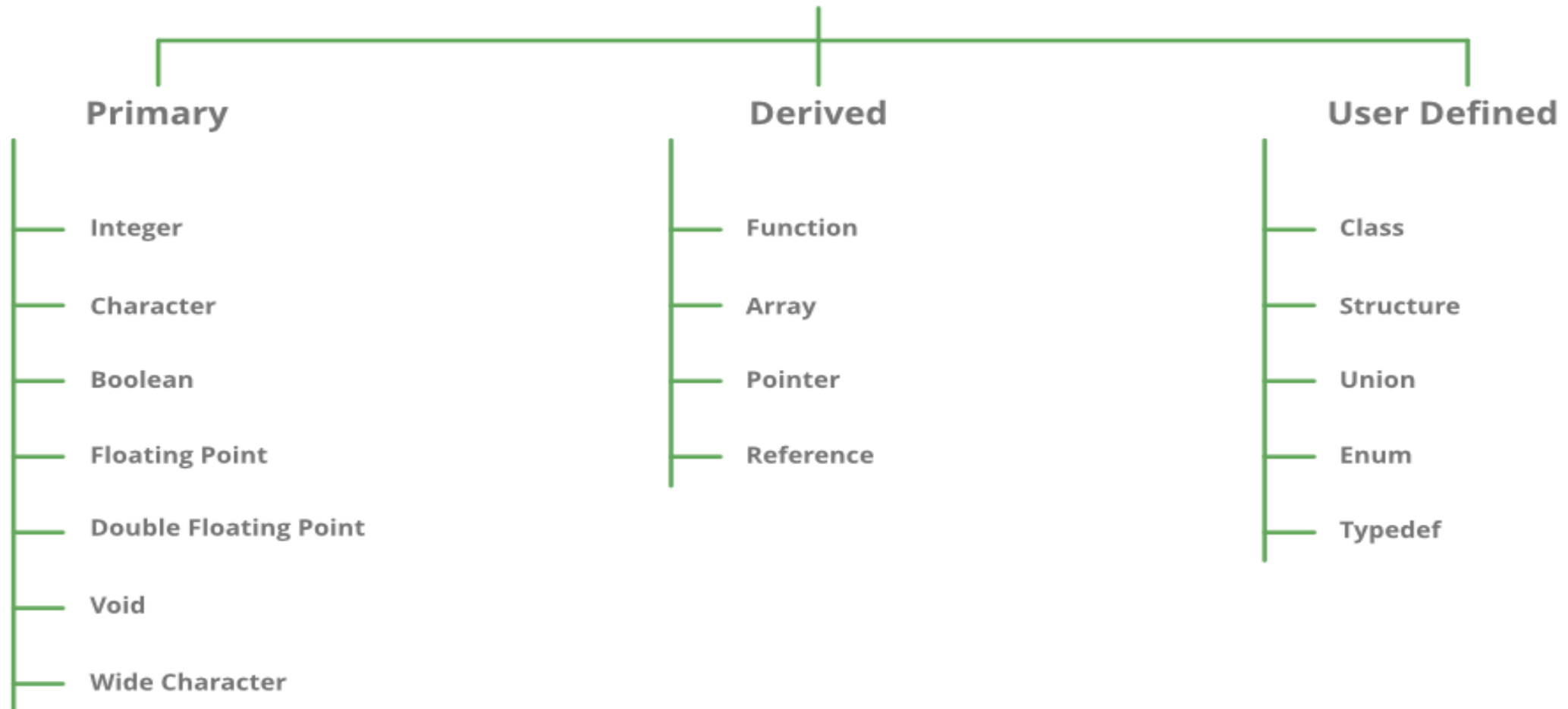


# C++ program life cycle

1. Write the code (computer program)
2. Compile it
3. Generates a program (e.g code.exe file on windows)




# DataTypes in C / C++



Type	Definition	Control Character	Limits
<b>int</b>	Integer		-2147483648 to 2147483647
<b>short</b>	Short Integer		-32768 to 32767
<b>long</b>	Long Integer	l or L	-2147483648 to 2147483647
<b>float</b>	Floating Decimal Number	f or F	1.17549e-038 to 3.40282e+038
<b>double</b>	Double Decimal Number		2.22507e-308 to 1.79769e+308
<b>long double</b>	Long Decimal Number		2.22507e-308 to 1.79769e+308
<b>char</b>	Character		-128 to 127
<b>unsigned int</b>	Unsigned Integer		0 to 4294967295
<b>unsigned short</b>	Unsigned Short Integer		0 to 65535
<b>unsigned long</b>	Unsigned Long Integer		0 to 4294967295
<b>unsigned char</b>	Unsigned Character		0 to 255
<b>bool</b>	True or False		True = 1 and False = 0



# 3. Basic Operations

		Operator	Type
<b>Binary Operator</b>		+, -, *, /, %	Arithmetic Operators
		<, <=, >, >=, ==, !=	Relational Operators
		&&,   , !	Logical Operators
		&,  , <<, >>, ~, ^	Bitwise Operators
		=, +=, -=, *=, /=, %=	Assignment Operators
<b>Unary Operator</b>	→	++, --	Unary Operator
<b>Ternary Operator</b>	→	?:	Ternary or Conditional Operator

# 4-Input and Output



- `cin>>` reads input from the user.
- `cout<<` outputs data to the console.

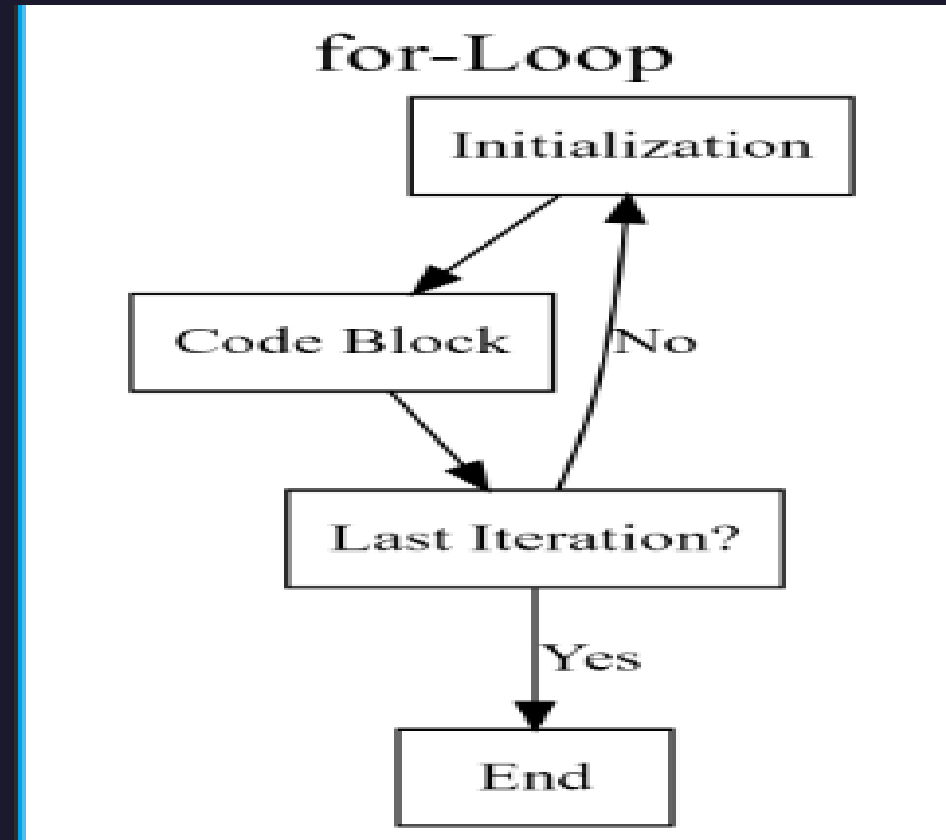
## 5-Conditional Statements :

```
if (/* condition */)
{
    /* code */
}
else{
    /* code */
}
```



# 6-loops

- `While( condition ) { // code }`
- `do{ /* code */ } while ( /* condition */ );`
- `for ( int i = 0 ; /* condition */ ; i++ ){`  
    `/* code */`     `}`



# 7. Functions

## C-Runtime stack usage

```
int foo(int,int) ;
main ( )
{
    int arr[1024] ;
    foo (10,11) ;
}
int foo (int a, int b)
{
    int arr1[1024] ;
    int arr2[1024] ;
    return (a+b) ;
}
```

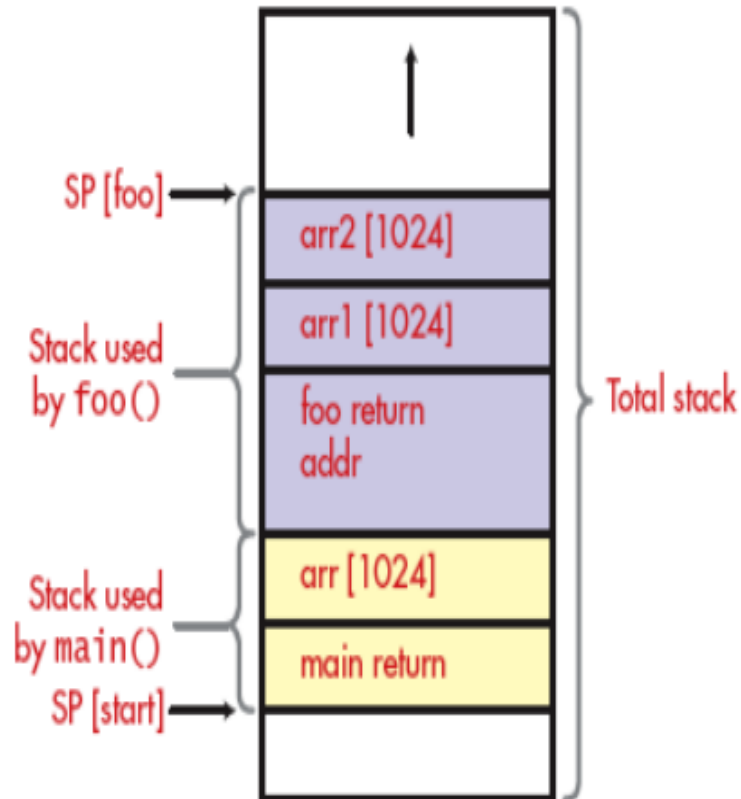


Diagram illustrating the components of a function declaration:

```
return type      Parameter Type
  ↑              ↑
int sum ( int a , int b );
  ↓              ↓              ↓
Function Name   Parameter Name Ending Statement Semicolon
```

Thank you

