/Basic Skeleton Code of C++

```
#include<iostream>

int main(){
std::cout<<"Hey! I Am Shashank"<<"\n";  // "\n" Prints in the new line
std::cout<<"Hey I have started Learning C++ and I'm proud of myself";
std::cout<<"Hey I have started Learning C++"<<std::endl<<"Hey Shubham""<<std::endl;
cin>>x>>y;
cout<<"The value of x is"<<x<<"The value of y is:"<<y<<endl;
return 0;
}
```

You can use std::endl instead of "\n"

Comments in C++

```
//Single line

/* Multiple
Line
*/
```

All the Libraries,header files are included in
#include<bits/stdc++.h>

Datatypes

Integer- int x;

String- string s;

String s1;

Cin >>S1;
Cout<<s1;

Input String : "Hey Wassup"
Output String : "Hey"

So,to take multiple strings We use the below code

```
String s1,s2;
cin>>s1>>s2;
cout<<s1<<" "<<s2;
```

```
Float x;
Double y;
Long z;
Long long z;
```

<u>To get an entire line of characters</u>

```
String str;
Getline(cin,variable_name);
cout<<str;
```

<u>Characters-Char</u>

```
Char ch;
cin>>ch;
cout<<ch;
```

```
Char ch='h';
cout<<ch;
```

Single inverted comma for Char.
Double inverted comma for String.

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	8bytes	-9223372036854775808 to 9223372036854775807
signed long int	8bytes	same as long int
unsigned long int	8bytes	0 to 18446744073709551615
long long int	8bytes	-(2^63) to (2^63)-1
unsigned long long int	8bytes	0 to 18,446,744,073,709,551,615
float	4bytes	
double	8bytes	
long double	12bytes	
wchar_t	2 or 4 bytes	1 wide character

If-Else loop

```
#include<bits/stdc++.h>
Int main(){

if(Condition){
Statement
}
Else(Condition){
Statement
}
Return 0;
}
```

If-Else if loop

```
#include<bits/stdc++.h>
Int main(){

if(Condition){
Statement
}
Else if(Condition){
Statement
}
Return 0;
}
```

If-Else if-elseloop

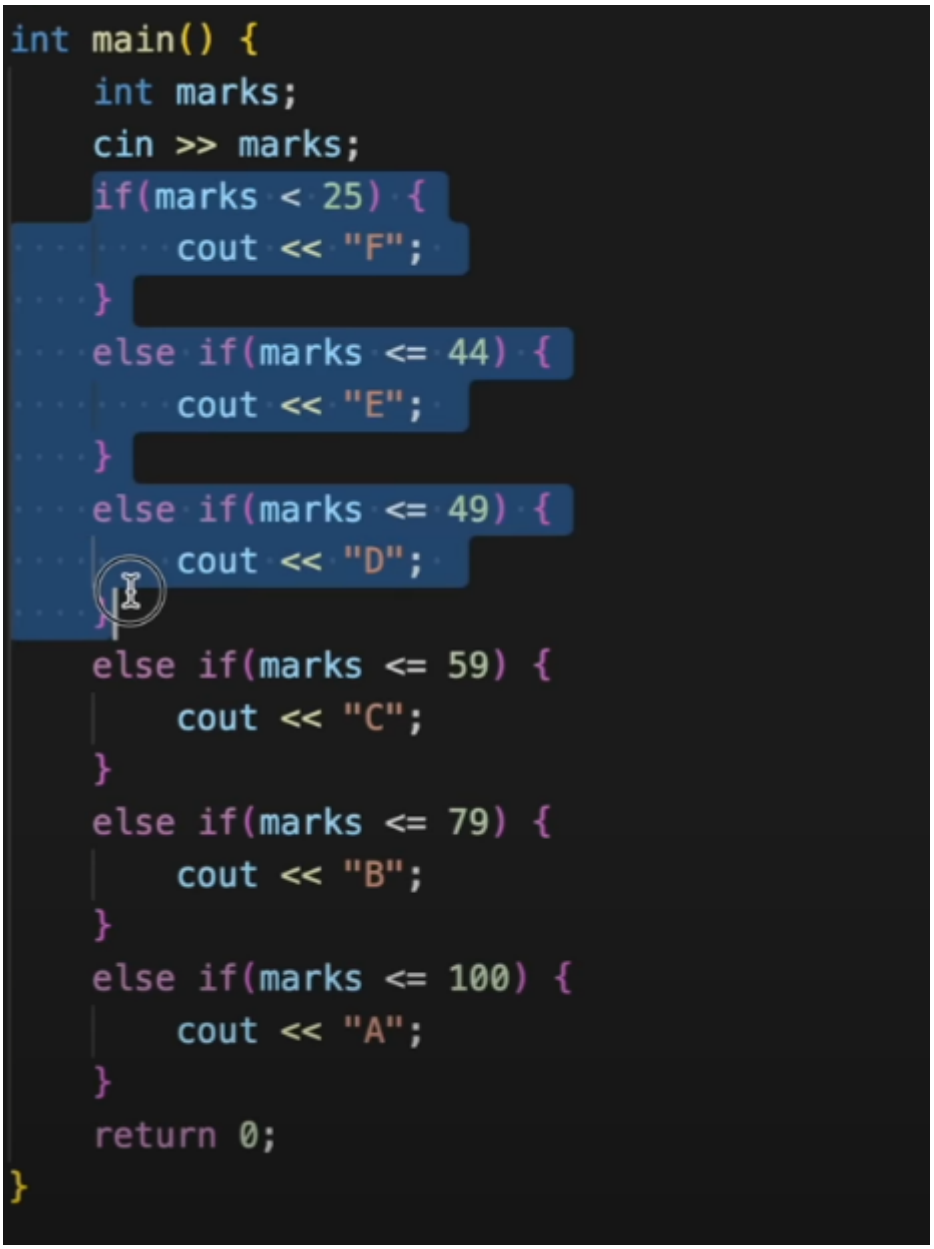
```
#include<bits/stdc++.h>
Int main(){

if(Condition){
Statement
}
Else if(Condition){
Statement
}
Else(Condition){
Statement
}

Return 0;
}
```

```
/*
A school has following rules for grading system:
a. Below 25 - F
b. 25 to 44 - E
c. 45 to 49 - D
d. 50 to 59 - C
e. 60 to 79 - B
f. 80 to 100 - A
Ask user to enter marks and print the corresponding grade.
*/
```

Very bad Code

```
int main() {
    int marks;
    cin >> marks;
    if(marks < 25) {
        cout << "F";
    }
    if(marks >= 25 && marks <= 44) {
        cout << "E";
    }
    if(marks >= 45 && marks <= 49) {
        cout << "D";
    }
    if(marks >= 50 && marks <= 59) {
        cout << "C";
    }
    if(marks >= 60 && marks <= 79) {
        cout << "B";
    }
    if(marks >= 80 && marks <= 100) {
        cout << "A";
    }
    return 0;
```

Good Code



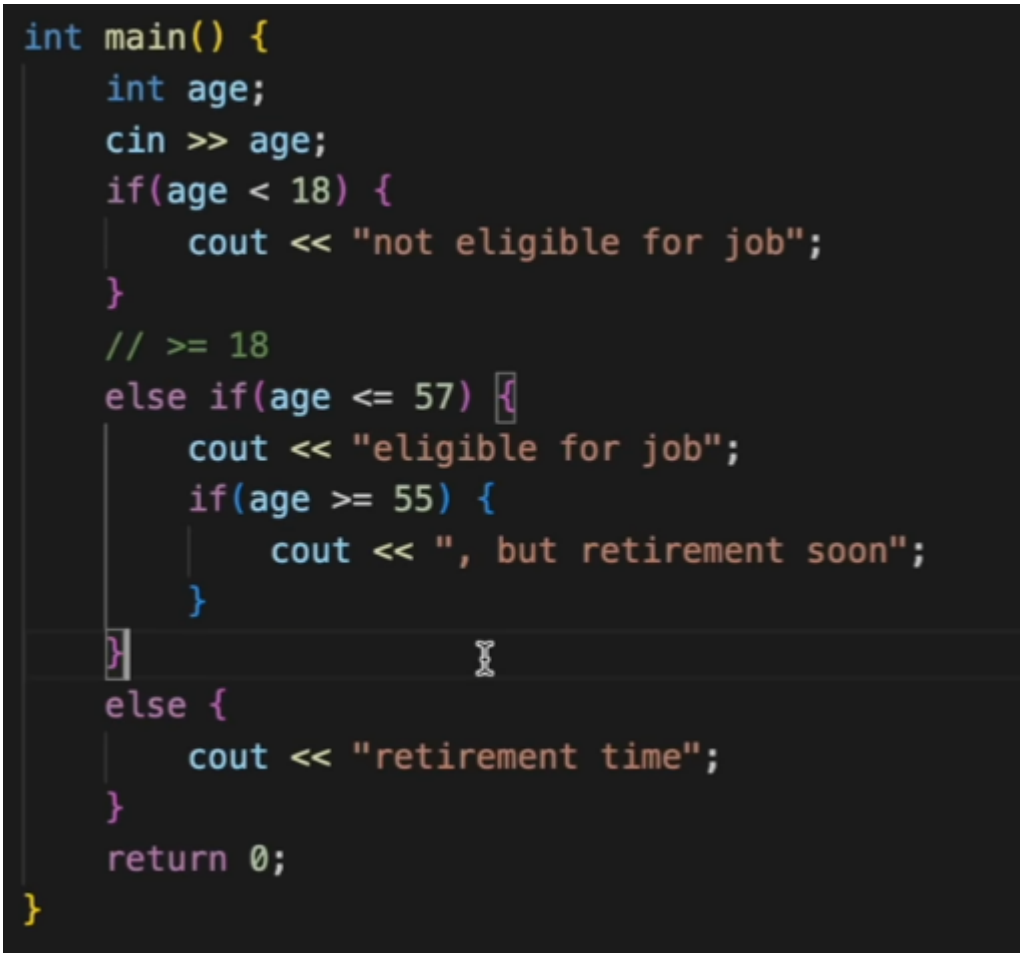
```
int main() {
    int marks;
    cin >> marks;
    if(marks < 25) {
        cout << "F";
    }
    else if(marks <= 44) {
        cout << "E";
    }
    else if(marks <= 49) {
        cout << "D";
    }
    else if(marks <= 59) {
        cout << "C";
    }
    else if(marks <= 79) {
        cout << "B";
    }
    else if(marks <= 100) {
        cout << "A";
    }
    return 0;
}
```

```
/*
Take the age from the user and then decide accordingly
1. If age < 18,
   print-> not eligible for job
2. If age >= 18,
   print-> "eligble for job"
3. If age >= 55 and age <= 57,
   print-> "eligible for job, but retirement soon."
4. If age > 57
   print-> "retirement time"
*/
```

Good Code

```
int main() {
    int age;
    cin >> age;
    if(age < 18) {
        cout << "not eligible for job";
    }
    else if(age <= 54) {
        cout << "eligible for job";
    }
    else if(age <= 57) {
        cout << "eligible for job, but retirement soon";
    }
    else {
        cout << "retirement time";
    }
    return 0;
```

Good Code using Nested If



```
int main() {
    int age;
    cin >> age;
    if(age < 18) {
        cout << "not eligible for job";
    }
    // >= 18
    else if(age <= 57) {
        cout << "eligible for job";
        if(age >= 55) {
            cout << ", but retirement soon";
        }
    }
    else {
        cout << "retirement time";
    }
    return 0;
}
```

Switch Statements

```
#Include<bits/stdc++.h>
Int main()
{
Int day;
cin>>day;
switch(day){

Case 1:
Condition
break;
Case2:
Condition
break;

Case 3:
Condition
break;

Case n:
Condition
Break;

Default:
Invalid
}

Return 0;
}
```

When the condition inside the case is executed,it automatically breaks out of the switch case loop,so anything extra in the switch case loop will not be executed after break

You can give default to deal with invalid inputs

Arrays

Datatype arrayname[n];
Example : int arr[5];

cin>>arr[0]>>arr[1]>>arr[2]>>arr[3]>>arr[4];

Arrays store values of the same datatype
Can be modified

Operations can be done on the values

Multi Dimensional Array

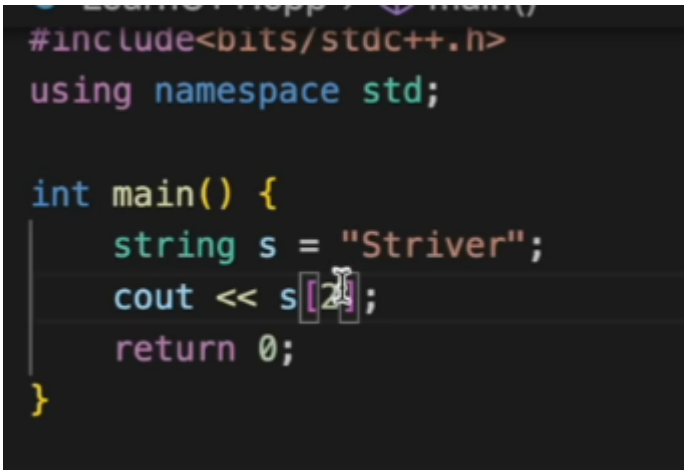
Int arr[3][5];
Int arr[row][column];

2D Arrays can be used in solving a lot of Graph,matrix problems
Arrays have 0 based index

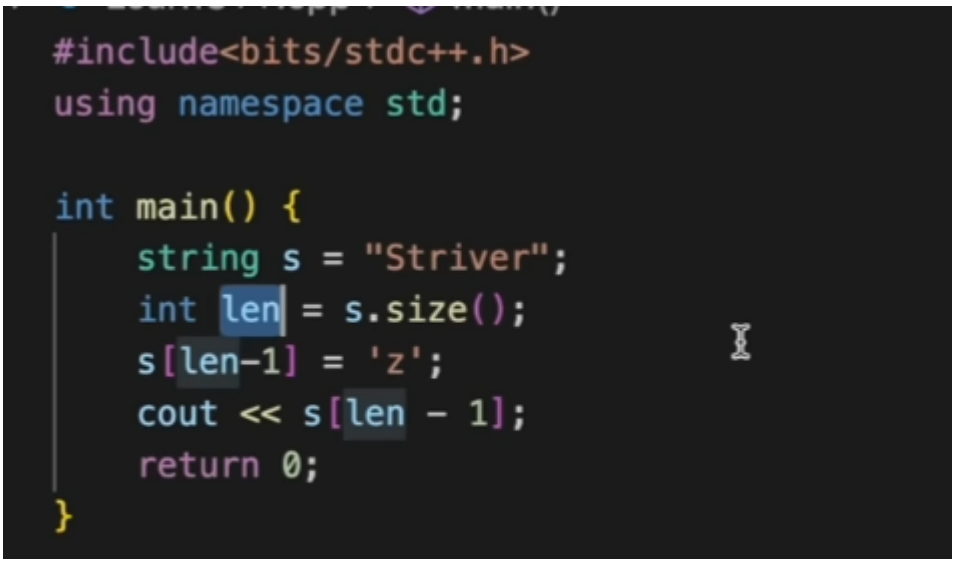
Strings

Strings is a combination of characters
We can access characters in a string like arrays

If we want to assign value to a specific position,then we can use single inverted commas to assign value,as we are changing the characters



```cpp
#include<bits/stdc++.h>
using namespace std;

int main() {
    string s = "Striver";
    cout << s[2];
    return 0;
}
```



```cpp
#include<bits/stdc++.h>
using namespace std;

int main() {
    string s = "Striver";
    int len = s.size();
    s[len-1] = 'z';
    cout << s[len - 1];
    return 0;
}
```

For Loops

for(Variable Inititialization;Condition;Increment/Decrement or change the value of the variable in any manner)

Example:

```
 for(i=1;i<=10;i){
cout<<"Hey! I am Shashank"<<endl;
}
```

If the variable is declared only in the for loop,it's scope stays only inside the loop.
To print the variable,declare the variable outside the for loop and initialise it in the loop.

Nested for loops are used for Patterns

While Loop

Initialization is the first thing in While loop,outside the while loop
Condition is inside the brackets

Statements inside the loop

Example:

```
Int i=1;
while(i<5){
cout<<"Hey!"<<endl;
}
```

Do while Loop

Atleast execute once
Execute even if the condition is false

```
Do{
cout<<"Hey!"<<endl;
i=i+1
}while(i<5);
```

For loop	While loop
Initialization may be either in loop statement or outside the loop.	Initialization is always outside the loop.
Once the statement(s) is executed then after increment is done.	Increment can be done before or after the execution of the statement(s).
It is normally used when the number of iterations is known.	It is normally used when the number of iterations is unknown.
Condition is a relational expression.	Condition may be expression or non-zero value.
It is used when initialization and increment is simple.	It is used for complex initialization.
For is entry controlled loop.	While is also entry controlled loop.
for (init ; condition ; iteration) { statement(s); }	while (condition) { statement(s); }
used to obtain the result only when number of iterations is known.	used to satisfy the condition when the number of iterations is unknown

Functions

```
// Functions are set of code which performs something for you
// Functions are used to modularise code
// Functions are used to increase readability
// Functions are used to use same code multiple times
```

```
// void -> which does not returns anything
// return
// parameterised
// non parameterised
```

Void Non Parametrized - Not returning anything

```
void printName() {
    cout << "hey Striver!";
}
int main() {

    return 0;
}
```

Void Parametrized

```
void printName(string name) {
    cout << "hey " << name << endl;
}
int main() {
    string name;
    cin >> name;
    printName(name);

    string name2;
    cin >> name2;
    printName(name2);
    return 0;
}
```

You are supposed to call all the functions and declare the variables in the Main Function

Return-Function

```
// Take two numbers and print its sum
int sum(int num1, int num2) {
    int num3 = num1 + num2; // 5 + 6 = 11
    return num3;
}

int main() {
    int num1, num2;
    cin >> num1 >> num2;
    int res = sum(num1, num2);
    cout << res;
    return 0;
}
```

Inbuilt Functions(Explore this more)

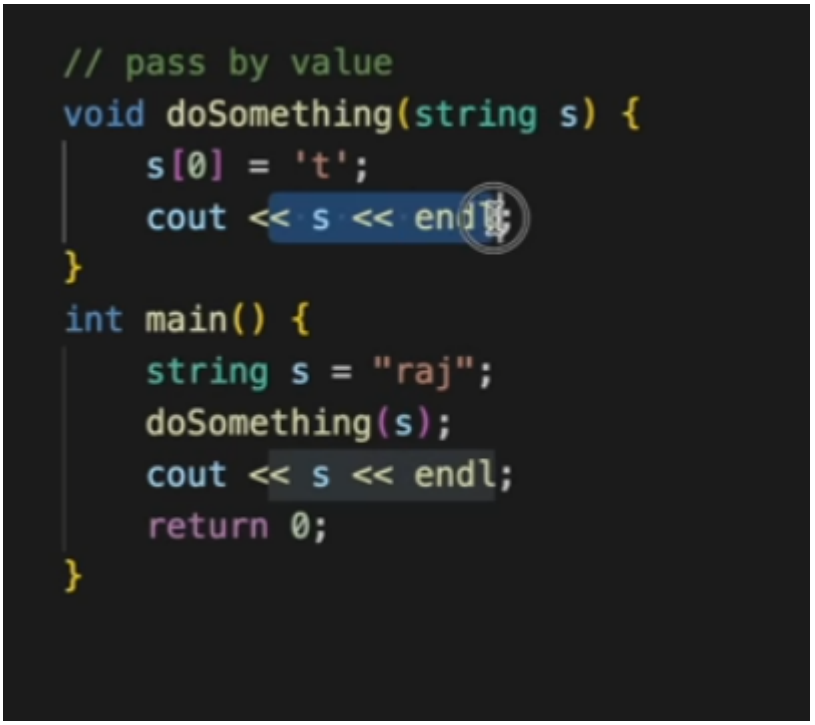
```
int main() {
    int num1, num2;
    cin >> num1 >> num2;
    int minimum = min(num1, num2);
    cout << minimum;
    return 0;
}
```

Same,with respect to max

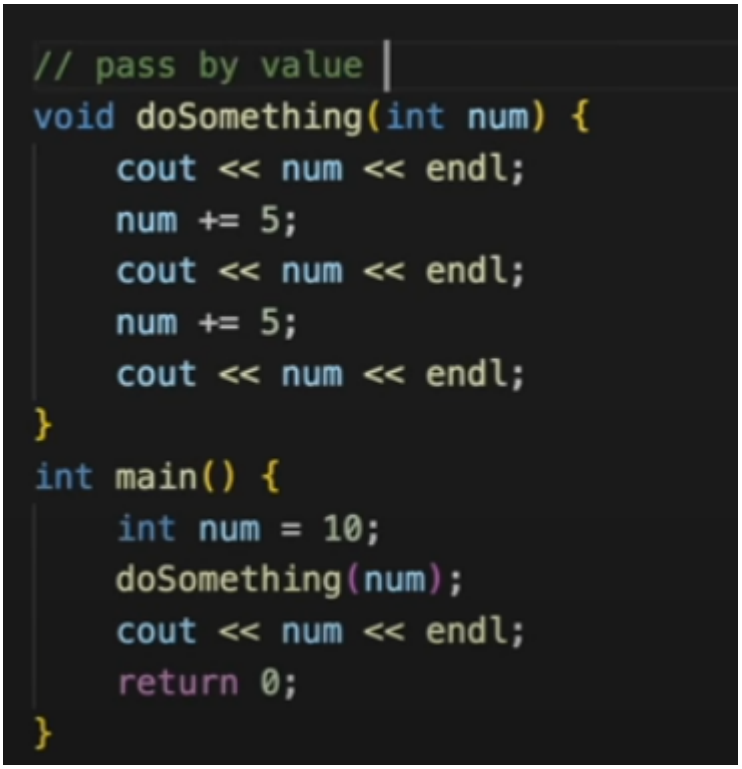
If you are writing a return function,make sure to write return,otherwise it'll return a garbage value

- **<u>Pass by value</u>-You are sending the copy of the original variable that is declared in the main function to the function,and the value associated with the variable is changed in the function**

- **But the original variable value in the main function,still remains the same**



```
// pass by value
void doSomething(string s) {
    s[0] = 't';
    cout << s << endl;
}
int main() {
    string s = "raj";
    doSomething(s);
    cout << s << endl;
    return 0;
}
```



```
// pass by value
void doSomething(int num) {
    cout << num << endl;
    num += 5;
    cout << num << endl;
    num += 5;
    cout << num << endl;
}
int main() {
    int num = 10;
    doSomething(num);
    cout << num << endl;
    return 0;
}
```

<u>Pass by Reference-</u> When you are giving parameters to the function,like in the below example,if it was a case of Pass by value,you would have used <u>"string s"</u>
In Pass By reference,you will be using the same thing, "String S",but with a &

That is "String &S"

That is,the address of the variable of the variable that is declared in the main function is used and not the copy of it,like that of Pass by Value

```
// pass by reference
void doSomething(string &s) {
    s[0] = 't';
    cout << s << endl;
}
int main() {
    string s = "raj";
    doSomething(s);
    cout << s << endl;
    return 0;
}
```

- **Pass by Reference-Refers to the main variable value**
- **Only Arrays are default passed by Reference,without using the &**
- **The rest are default Pass by Value**

```
// pass by reference
void doSomething(int arr[], int n) {
    arr[0] += 100;
    cout << "Value inside function: " << arr[0] << endl;
}
int main() {
    int n = 5;
    int arr[n];
    for(int i = 0;i<n; i = i + 1) {
        cin >> arr[i];
    }

    doSomething(arr, n);

    cout << "Value inside int main: " << arr[0] << endl;
    return 0;
}
```