```
// C++ Basic Structure & Cheat-Sheet By HackStark Version=1
// #include <iostream>
// // #include <iostream>
// using namespace std;
// int main()
// {
//     // cout << "Hello World"<<endl;
// cout << "\n \t Hello World";
// cout << "\n Hello World \t ";
// Or You Can Use Pause So
// {
//     cout << "Hello World  ";
//     system("pause");
// }
// int a,b,c;
// short sa=54;
// short sb=54;
// short Sb=4;
// cout<<sa<<endl;
// cout<<sb<<endl;
// cout<<Sb;
// int mark=1;
// // cout<<mark;
// float score=44.5555555;
// cout<<"score is "<<scor;
//      The sum Of Two Numbers
//     int a, b;
//     cout<<"Enter first Number"<<endl;
//     cin>>a;
//     cout<<"Enter second Number"<<endl;
//     cin>>b;

//     cout<<"The a+b is equal to  "<<a + b<<endl;
//     cout<<"The a-b is equal to  "<<a - b<<endl;
//     cout<<"The a*b is equal to  "<<a * b<<endl;
//     cout<<"The a/b is equal to  "<<a / b<<endl;
// If we need reminder in division the we can use a%b

// int age;
// cout << "Enter Yor Age?" << endl;
// cin >> age;
// switch (age)
// {
// case 12:
//     cout << "You Are 12 Years OLD";
//     break;

// case 18:
//     cout << "You Are 18 Years OLD";
//     break;
// default:
// cout<<"You Are Neither 12 nor 18 Years OLD";
//     break;
// }

// if (age>150 || age<1)
// {
//     cout<<"Invalid Age";
// }
//  else if (age>=18)
// {
//     cout<<"You Can Vote";
// }
// else {
//     cout<<"You Can't Vote";
// }
// For While Loop
// int index = 0;
// while (index < 31)
// {
//     // cout << "\n The  Index number  is Here  \t " << index;
//     cout << " The  Index number  is Here " << index << endl;
//     index = index + 1;
// }

// For Wrong While Loop When Condition Is False Nothing Will Show in Output
// int index = 0;
// while (index > 31)
// {
//     // cout << "\n The  Index number  is Here  \t " << index;
//     cout << " The  Index number  is Here " << index << endl;
//     index = index + 1;
// }

//  For Do While Loop (It will must run at once Don Not Matter Condition is True Or
False)

// int index = 0;
// do
// {
//     cout << " The  Index number  is Here " << index << endl;
//     index = index + 1;
// } while (index > 345);
// For Loop Starts From Here (If Condition is True Then Output Will Print)
// for (int i = 0; i < 7; i++)
// {
//     cout << "The Value of i is " << i << endl;
// }

// return 0;
// }

// Starting Functions From Scratch (Built-in Functions and  User-Defined
Functions )
// Let Function in MATH is { f(x)= x^3 +3 } This Was Example of Function

// #include <iostream>
// using namespace std;
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
// int main()
// {
//     int a, b;
//     cout << "Enter first Number" << endl;
//     cin >> a;
//     cout << "Enter second Number" << endl;
//     cin >> b;
//     cout << "The Function Returned is " << add(a, b);
//     return 0;
// }
// Starting Arrays From Scratch (An Array is a Collection of Elements of the Same
Type Placed in contiguous
// memory locations that can be individually referenced by using an index to a
unique identifier. )
// (An array is a data structure that contains a group of elements.)
// Arrays are used to store multiple values in a single variable, instead of
declaring separate variables for each value.
// ( In C++ programming language we do have mainly two types of variables: Single
Dimensional Arrays and multidimensional Arrays)
// Single Dimensional Array:- Example 1

// #include <iostream>
// using namespace std;
// Let Function in MATH is { f(x)= x^3 +3 } This Was Example of Function
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
// int main()
// {
//    //Array index 0, 1 ,2              OR Line 140 Can Be: int arr[] = {2, 4, 7};
//     int arr[] = {2, 4, 7};
//     cout<<arr[1];

//     return 0;
// }
//

// Single Dimensional Array:- Example 2
// Going To Hard  Coding
// Single Dimensional Array is Used To Represent A List
// First This Code add a function and then arrray is added in function in the array
For Loop is used Twice
// #include <iostream>
// using namespace std;
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
// int main()
// {
//    //Array index 0, 1 ,2              OR Line 159 Can Be: int arr[] = {2, 4, 7};
//     int arr[] = {2, 4, 7};
//     // cout << arr[1];
//     int marks[7];
//     for (int i = 0; i < 6; i++)
//     {
//         cout << "Enter The Marks of " << i << " th student" << endl;
//         cin >> marks[i];
//     }
//     for (int i = 0; i < 6; i++)
//     {
//         cout << "Marks of " << i << " th student is " << marks[i] << endl;
//     }
//     return 0;
// }


// Two  Dimensional Array Starts From Here:
//  Starting 2 Dimensional Arrays From Scratch:-
// #include <iostream>
// using namespace std;
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
// int main()
// {
//     int arr2d[3][4] = {
//         {1, 2, 3, 4},
//         {5, 6, 7}};
//     for (int i = 0; i < 2; i++)
//     {
//         for (int j = 0; j < 3; j++)
//         {
//             cout << "The Value at " << i << ", " << j << " is " << arr2d[i][j]
<< endl;
//         }
//     }

//     return 0;
// }


// // Starting Type-Casting From Scratch:-
// #include <iostream>
// using namespace std;
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
// int main()
// {

//     int a = 344;
//     float r = 89.93;
//     cout << (float)a / 24<<endl;
//     cout << (int) r;

//     return 0;
// }
// Starting String From Scratch:-
// Strings are used for storing text.A string variable contains a collection of
characters surrounded by double quotes/
// #include <iostream>
// #include <string>
// using namespace std;
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
// int main()
// {
//     string name = "Talha";
//     cout << "The Name is " << name << endl;
//     cout << "The Length of Name is " << name.length() << endl;
//     cout << "The First Letter of Name is " << name.substr(0, 1) << endl;
//     cout << "The First Two Letters of Name is " << name.substr(0, 2) << endl;
//     cout << "The First Three Letters of Name is " << name.substr(0, 3) << endl;
//     cout << "The First Four Letters of Name is " << name.substr(0, 4) << endl;
//     // For Better Understand See OutPut Carefully
//     cout << "The second & Third Letter of Name is " << name.substr(1, 2) <<
endl;
//     cout << "The second , Third & Fourth Letter of Name is " << name.substr(1,
3) << endl;
//     cout << "The second , Third , Fourth & Fifth Letter of Name is " <<
name.substr(1, 4) << endl;
//     cout << "The Third , Fourth & Fifth Letter of Name is " << name.substr(2, 3)
<< endl;

//     return 0;
// }


// Starting Pointers From Scratch:-
// The variable that stores the address of another variable is Called Pointers.
// Pointers are a very powerful feature of the language that has many uses in lower
level programming.
// Pointers is uded in Dynamic Memory Allocation.
// #include <iostream>
// #include <string>
// using namespace std;
// int add(int a, int b) {
//     int c;
//     c = a + b;
//     return c; }
// int main() {
//     int a = 23;
//     int *ptra;
//     // At int the float can also be use
//     ptra = &a;
//     cout << "The Value of a is " << a << endl;
//     cout << "The Value of a is " << *ptra << endl;
//     cout << "The adress of a is " << &a << endl;
//     cout << "The adress of a is " << ptra << endl;
//     return 0;
// }
// Starting Object-Oriented Programming From Scratch:-
// OOP is Like a Template
// OOP is a Method of programing in which we make classes , Objects and Templates
// searchapparchitecture.techtarget.com OOP: is a computer programming model that
organizes software design around
// data, or objects, rather than functions and logic. An object can be defined as a
data field that has unique attributes and behavior.
// #include <iostream>
// #include <string>
// using namespace std;
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
//  class Employee {
//      public:
//      string name;
//      int salary;

//  };
//  int main()
//  {
// Employee Tal;
// Tal.name = "Talha";
// Tal.salary = 300;
// cout<<"The Name of Employee is "<<Tal.name<<" and his salary is
"<<Tal.salary<<"$"<<endl;

// return 0;

//  }


// 2nd Method of Previous OOP program
// #include <iostream>
// #include <string>
// using namespace std;
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
// class Employee
// {
// public:
//     string name;
//     int salary;
//     void printDetails()
//     {
//         cout << "The Name of Employee is " << this->name << " and his salary is
" << this->salary << "$" << endl;
//     }
// };
// int main()
// {
```

```
//     Employee Tal;
//     Tal.name = "Talha";
//     Tal.salary = 300;
//     Tal.printDetails();

//     return 0;
// }


// Constructor in OOP of C++ : Using 2nd Method of Previous OOP program

// #include <iostream>
// #include <string>
// using namespace std;
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
// class Employee
// {
// public:
//     string name;
//     int salary;
//     Employee(string name, int salary)
//     {
//         this->name = name;
//         this->salary = salary;
//     }

//     void printDetails()
//     {
//         cout << "The Name of Employee is " << this->name << " and his salary is
" << this->salary << "$" << endl;
//     }
// };
// int main()
// {
//     Employee Tal("Talha Constructor", 330);

//     Tal.printDetails();

//     return 0;
// }





// OOP With Private Variables
// Constructor in OOP of C++ : Using 2nd Method of Previous OOP program

// #include <iostream>
// #include <string>
// using namespace std;
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
// class Employee
// {
// public:
//     string name;
//     int salary;
//     Employee(string name, int salary, int secretPassword)
//     {
//         this->name = name;
//         this->salary = salary;
//         this->secretPassword = secretPassword;
//     }

//     void printDetails()
//     {
//         cout << "The Name of Employee is " << this->name << " and his salary is
" << this->salary << "$" << endl;
//     }
//     void getSecretPassword()
//     {
//         cout<<"The Secret Password of Employee is "<<this->secretPassword;
//     }

// private:
//     int secretPassword;
// };
// int main()
// {
//     Employee Tal("Talha Constructor", 330, 995544231);

//     Tal.printDetails();
//     // This line Will Not Give secretPassword Because it is private
//     // cout << Tal.secretPassword;
//     // Then From inside the function is
//     Tal.getSecretPassword();

//     return 0;



// inheritance in c++
// Inheritance is one of the key features of Object-oriented programming in C++. It
allows us to create a new class
// (derived class) from an existing class (base class). The derived class inherits
the features from the base class
// and can have additional features of its own.
// OOP With Private Variables
// Constructor in OOP of C++ : Using 2nd Method of Previous OOP program

// #include <iostream>
// #include <string>
// using namespace std;
// int add(int a, int b)
// {
//     int c;
//     c = a + b;
//     return c;
// }
// class Employee
// {
// public:
//     string name;
//     int salary;
//     Employee(string name, int salary, int secretPassword)
//     {
//         this->name = name;
//         this->salary = salary;
//         this->secretPassword = secretPassword;
//     }

//     void printDetails()
//     {
//         cout << "The Name of Employee is " << this->name << " and his salary is 
" << this->salary << "$" << endl;
//     }
//     void getSecretPassword()
//     {
//         cout<<"The Secret Password of Employee is "<<this->secretPassword;
//     }

// private:
//     int secretPassword;
// };
// // starting inheritance
// class Programmer : public Employee
// {
//     public:
//     int errors;
// };
// int main()
// {
//     Employee Tal("Talha Constructor", 330, 995544231);

//     Tal.printDetails();
//     // This line Will Not Give secretPassword Because it is private
//     // cout << Tal.secretPassword;
//     // Then From inside the function is
//     Tal.getSecretPassword();

//     return 0;
// }






// // Thank You All
```