```cpp
#include <iostream>

#include <string>
//to use string in C++, we need to include string library

#include <cmath>
//library for more available math function

#include <fstream>
//file I/O stream

#include <vector>
using namespace std;

//create a class
class Car{
    public:
    //public members
        string brand;
        string model;
        int year;
        //some attributes

        Car(string b, string m, int y){
            brand = b;
            model = m;
            year = y;
        }
        //constructor


        void printCar(){
            cout << brand << model << year;
        }
        //some methods (method defined inside the class)

        int carYear();//getter
        //alternative way: method header inside the class

    private:
    //private members (by default, members are private)
        int days;

};

int Car::carYear(){
    return Car::year;
}
//finish the method implementation outside of the class



//inheritance
//class subClass: public superClass{...};
class truck: public Car{
    public:
        double truck_size;
} ;


class wheel{

};


//multiple inheritance
```

```cpp
class bike: public Car, public wheel{

};
//separate by commas

//recursion
int sumOfDigits(int number = 0){
    //default parameter value "  = *some value* "
    int sum = 0;
    if(number < 10){
        sum = number;
    }else{
        int onesDigit = number % 10;
        sum = onesDigit + sumOfDigits(number/10);
    }
}

//pass by reference
int sumOfArray(int* arr, int arrSize){
    int sum = 0;
    for(int i = 0; i < arrSize; i++){
        sum += *(arr + i);
    }
    return sum;
}

struct structureTemplate {
    int myAge;
    string myName;
    char myInitial;
};
//named structure
//declare a "structure type" outside of the main
//treat this kind of structure as "a new data type"

//for vector: pass by reference
void print_vector(vector<int> &vec){
    for(int i = 0; i < vec.size(); i++){
        cout << vec[i] << " ";
    }
}


int main() {
    cout << "Hello World!";
    //standard output

    const int myAge = 10;
    //a constant

    bool isVisited = false;
    //return value: 0 or 1 (logic value)

    cout << "I am " << myAge << " years old!";
    //For output stream concatenation (use "<<" to concatenate instead of "+")
    //for string concatenation (simply use "+")

    string userInput;
    //string type: lowercase "s"

    //cin >> userInput;
    //take user input, but only take the first token
    //in order to take an entire line, we need to do the following:
    //getLine(cin,userInput);


```

```
    string last_name = "Peng";
    string first_name = "Yusen";
    string fullName = first_name.append(last_name);
    //an alternative way to concatenate strings

    int pos = 0;
    cout << fullName[pos];
    //access strings just like arrays


    //C++ Math
    min(3,2);
    max(5,9);
    //these two functions is independent of the <cmath> library

    sqrt(16);
    round(2.3);
    log(10);//natural log
    pow(2,5);
    abs(-2);

    //C++ arrays
    int arr[10] = {1,2,3,4,5,6,7,8,9,10};
    //very similar to C language
    //the array's size can be omitted

    int arr_size = sizeof(arr) / sizeof(int);
    //determine the array's size -- very similar to C
    cout << sumOfArray(arr,arr_size);

    struct {
        int myAge;
        string myName;
        char myInitial;

    } myFirstStructure;
    //directly declare a structure in main and manipulate it
    myFirstStructure.myAge = 19;

    structureTemplate anInstance;
    //create an instance of "named structure" type
    anInstance.myAge = 21;
    anInstance.myInitial = 'P';

    cout << sumOfDigits(12345);

    ofstream my_file("filename.txt");
    my_file << "some text to write into file";
    my_file.close();
    //create and write a file


    string text;
    ifstream his_file("filename.txt");
    //while(getLine(his_file, text)){
    //    cout << text;
    //}
    his_file.close();


    //dynamic memory allocation
    //keyword: new
    //For a single variable
    int* ptr_int = new int;
    *ptr_int = 34;
    //store the value 5 in the heap
```

```
    printf("%d", *ptr_int);
    delete ptr_int;
    //explicitly delete

    //dynamic memory allocation for array
    int* anotherArray = new int[4];
    for(int i = 0; i < 4; i++){
        *(anotherArray + i) = 2*i;
        cout << *(anotherArray+i);
    }
    delete anotherArray;

    //vector
    vector<int> my_vector;
    my_vector.push_back(12);
    //add element at the very end

    cout << my_vector[my_vector.size()-1];

    my_vector.pop_back();
    //remove the last element


  return 0;
}
```