

```
#include <ctime> } (standard library header)  
#include <iostream> header.  
#include <string>  
using namespace std; for making code shorter  
library X is used to print output.  
#include <ctime> C++ header for working with time function
```

• This function used to generate a random number  
in this project. *header* *random header*

```
#include <iostream> C++ header for working with standard input & output  
>>> Input & Output functions.
```

• This function / header used to take an Input  
from the User and display output to the user.  
*header* *input & output*

```
#include <string> C++ header for working with a (Includes the string class to work with strings).
```

• This is used to String Variable to store the  
Character name. + *header* *string*

Using namespace std : allows the use of standard  
C++ library features like cout & cin. without std::  
prefix *header* *works* : *without std::*

Q. Difference Between #include<iostream> & using namespace std;  
Ans: std:: is not required if you bring iostream in your code  
and std:: is required if you want to use individual header files.

~~#include <iostream> (if required) to use the input & output function for all standard I/O library~~

~~<friend> shoring #~~

Using namespace std is optional but may add code by removing the need of prefix standard library names with std:: ~~<friend> shoring # without std::~~

With using namespace std

~~remove std:: before std::cout~~

~~#include <iostream>~~

~~using namespace std;~~

~~std::cout << "Enter two No's"; std::cin << "Enter two No's"~~

~~endl;~~

~~input is sent of base without std::~~

~~cin >> a; cout >> b; std::cin >> a; std::cout >> b;~~

~~cin >> b;~~

without using namespace std

~~of base without std::~~

~~#include <iostream>~~

~~std::cout << "Enter two No's"; std::cin >> a; std::cout >> b;~~

~~endl;~~

~~input is sent of base without std::~~

~~std::cin >> a; std::cout >> b;~~

~~std::cin >> b;~~

~~with friend yet shoring #~~

~~<friend> shoring #~~

~~(inside class name of endl)~~

protected

~~list name of members inside at base right~~

Symbols public :- ~~+ . scope resolution~~

~~protected :- #~~

~~but note for see private :- hta programming part~~

~~it is too much also it has will protect program~~

~~protected :- Inside class : Where members can be~~

~~accessed normally by all members of the class.~~

~~main point of Xcode of shoring # direct access without #~~

Derived class :- The member's can be accessed

by classes that inherit from the class

~~the member is defined (Base class).~~

outside class :- No The members cannot be accessed directly by non-member function or object. Unless they are friends of the class.

Q. What is Friend Function ?

Friend Function that is given special access to the private & protected members of a class even if it's not a member of that class.

A function as a friend by using the friend keyword inside the class definition.

Syntax : friend (function) return type function name (class name & reference);  
(e.g. friend ( Asha ) return type function name ( Asha ) ;  
me myfriend ( Asha ) myname = Asha  
Asha is my friend. She can access my home resource.

By code Example :-

(class) friend  
{  
 void fun ( )  
 {  
 cout << "Hello"  
 }  
}

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

class Sahana;

Class Rutuja : public Sahana

private:

int r.money = 20; // friend without barf  
friend void Suraksha(Rutuja, Sahana); // having  
friend void Suraksha(); // without barf

class Sahana : public Rutuja

private: int s.money = 5;

int money = 10;

friend void Suraksha(Rutuja, Sahana);

void Suraksha() {

Rutuja r; Sahana s;

cout << r.money << endl;

cout << s.money << endl;

cout << "Sum" << r.money + s.money << endl;

int main()

Sahana s;

Rutuja r;

Suraksha(r, s);

return 0;

```
string name;  
int health;  
int shield;
```

public:

```
Character(const string &NAME, int HEALTH  
int SHIELD) : name(NAME), health(HEALTH), shield(SHIELD)
```

{  
 // Initialization code here  
}

Destructor:- special member function that is called when an object is destroyed.

Virtual destructor :- Virtual destructor is a destructor that is declared with the Virtual keyword. (Inheritance).

Virtual destructor is declared with the Virtual keyword.

It ensures that when a derived class object is deleted through a base class pointer.

The destructor of the base class is called first followed by the destructor of the derived class.

Q Why virtual destructor? A. Virtual destructor ensures that all resources allocated to both the base and derived classes are properly released in case of memory leak.

member friend  
without func  
; before func

String getName() const {  
 return name; }

member function does not modify the state of the class  
Int object on which it's called. (CREATE THE  
CLASS) Building (HTM)

pure virtual function :- Is a function that is  
declared in a base class but not have to  
implement in that class.

It is meant to be overridden by derived classes.

Declaring a function or pure virtual is easy to  
create an abstract class.

Syntax :-

Virtual returnType functionName() = 0;

Q) What is Abstract class?  
A class that contains at least one pure virtual  
function is called an abstract class.

Virtual Functions :- It is a member function in a base  
class that you expect to override in derived  
classes.

When you are declaring member functions  
as virtual, you are informing the compiler  
that the function can be overridden and that  
it should use dynamic binding to determine  
which function to call based on actual type of  
object, not the type of the pointer or reference.

## Virtual Function ()

Has to be Implemented  
in the base class.

Allows derived classes to  
override and provide  
new implementation.

## hidden Pure Virtual Function.

Does not have an implementation in the base class.

Defines an interface that  
derived classes must  
implement.

(O → Nor.) + 3.13

; ((P → O bmr) - c) - hide = hide  
(S C I O)

(as bmr) +

↓ bmr → derived class  
↓ hide → base class

↓ bmr + derived = derived

↓ O - hide

(PF → Nor.) + 3.13

; ((d1 = (1 bmr + p) - hide) = hide  
- (p = S1, O)) -

P = O = (2.1.1) + P

P = O = (1.1.1) + P

P = O = (2.1.1) + P

P = O = (S1.1) + P

P = P = (2.1.1) + P

void void getAttacked() {

    int roll = rand(0, 100); // to generate a random integer.  
    if (roll == 0)  
        health = 0; // health  
    else if (roll <= 50)

        shield = shield - (2 - (rand() % 4));  
        if (shield < 0)

            health += shield;

        shield = 0;

    else if (roll <= 79)

        health = health - (4 + rand(0, 5));  
        if (roll <= 1)

$$4 + (0 \cdot 1 \cdot 5) = 0 = 4$$

$$4 + (1 \cdot 1 \cdot 5) = 1 = 5$$

$$4 + (2 \cdot 1 \cdot 5) = 2 = 6$$

$$4 + (3 \cdot 1 \cdot 5) = 3 = 7$$

$$4 + (4 \cdot 1 \cdot 5) = 4 = 8$$

PF ..... 82 48 10

int roll = rand(0, 100); // Divide by 100.

which results in a value 0 and 99.

100% chance: 100% fail

These values are stored in variable roll.

(roll > 100) == 1

if (roll == 0)

The character was hit in a special

health = 0;

way.

else if (roll <= 50)

This condition checks if roll is between 0 and 50

This represents a range where the character takes damage but not severe at a critical hit.

Shield = Shield - (2 - (rand(0, 4)));

100% chance: 100%

(roll > 100) == 1

This line modifies the shield value.

rand(4) generates a random number 0, 1, 2, 3

0 1 2 3

2 - 0 = 2

(or 2 + 1 - 1 + 1)

2 - (rand(0, 4))

2 - 2 = 0

(or 2 + 1 - 1 + 1 - 1)

2 - 0 = 2

2 - 1 = 1

2 - 2 = 0

2 - 3 = -1

2 / 0 = 12

2 / -1 = -12

2 / 2 = 1

if (shield < 0)

(roll > 100) == 1

health = health + shield;

shield = 0;

51 52 53 ..... 79

void getAttacked (1) void dealDamage = Her. + 0.1  
if brr < 0.110V > 0.7 shield damage

int roll = rand (0 % 100);  
Max shield of bird is 110V - 100

if (roll < 15) (o = 100) {

if (c + shield > maxShield) { o = 100; }

shield = maxShield; (o2 = 100) {

else if (c + shield < maxShield) { o = 100; }

roll = rand % 100;  
if ((c + o) / brr) - s) - health = bird

if (roll < 65) {

else if (c + shield < maxShield) { o = 100; }

shield = shield - (1 + rand (0.1 - 0.2));

c = o - b

else if (shield < 0) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }

else if (c + shield < maxShield) { o = 100; }