BATCH

RECURSION

C AND C++

PROGRAMMING MASTERCLASS

AWESH ISLAM
BUET, CSE

C++ Class-03

SHAROARE HOSAN EMON
BME , BUET

আমাদের সবগুলো ক্লাস দেখার জন্য ভিজিট করো

https://www.hsccrackers.com/

SCAN ME

A Closer look into Classes

Constructor & Destructor

```cpp
#include<iostream>
using namespace std;
class myClass{
    int a;
public:
    myClass();
    ~myClass();
    void show();
};
myClass::myClass(){
    cout<<"Constructing..."<<endl;
    a = 10;
}
myClass::~myClass(){
    cout<<"Destructing..."<<endl;
}
void myClass::show(){
    cout << a <<endl;
}
int main(){
    myClass ob;
    ob.show();
    return 0;
}
```

```
Constructing...
10
Destructing...
Program ended with exit code: 0
```

Constructor & Destructor

```
20  int main(){
21      for(int i = 0;i < 5;i++){
22          myClass ob;
23          ob.show();
24      }
25      return 0;
26  }
```

```
Constructing...
10
Destructing...
Constructing...
10
Destructing...
Constructing...
10
Destructing...
Constructing...
10
Destructing...
Constructing...
10
Destructing...
Program ended with exit code: 0
```

Stack Class

```cpp
#include<iostream>
using namespace std;
#define SIZE 10
class stack{
    char stck[SIZE];
    int tos;
public:
    stack();
    void push(char c);
    char pop();
};
stack::stack(){
    cout << "Constructing stack..." <<endl;
}
void stack::push(char c){
    if(tos == SIZE){
        cout<<"Stack is full."<<endl;
        return;
    }
    stck[tos] = c;
    tos++;
}
char stack::pop(){
    if(tos==0){
        cout<<"Stack is empty"<<endl;
        return 0;
    }
    tos--;
    return stck[tos];
}
```

```cpp
int main(){
    stack s1,s2;
    s1.push('a');
    s2.push('x');
    s1.push('b');
    s2.push('y');
    s1.push('c');
    s2.push('z');
    for(int i = 0;i < 3;i++) cout<<"Pop s1: " << s1.pop() <<endl;
    for(int i = 0;i < 3;i++) cout<<"Pop s2: " << s2.pop() <<endl;
    return 0;
}
```

```
Constructing stack...
Constructing stack...
Pop s1: c
Pop s1: b
Pop s1: a
Pop s2: z
Pop s2: y
Pop s2: x
Program ended with exit code: 0
```

Task

Implement a queue class by yourself

Example (String type)

```
4  #define SIZE 255
5  class strtype{
6      char *p;
7      int len;
8  public:
9      strtype();
10     ~strtype();
11     void set(char *ptr);
12     void show();
13 };
14 strtype::strtype(){
15     p = (char *) malloc(SIZE);
16     if(!p){
17         cout<<"Allocation Error\n";
18         exit(1);
19     }
20     *p = '\0';
21     len = 0;
22 }
23 strtype::~strtype(){
24     cout<<"Freeing..."<<endl;
25     free(p);
26 }
27 void strtype::set(char *ptr){
28     if(strlen(ptr) >= SIZE){
29         cout<<"String is too big"<<endl;
30         return;
31     }
32     strcpy(p, ptr);
33     len = (int)strlen(p);
34 }
35 void strtype::show(){
36     cout<< p << " - length: " << len <<endl;
37 }
```

```
38 int main(){
39     strtype s1,s2;
40     s1.set("This is a test.");
41     s2.set("I like C++.");
42     s1.show();
43     s2.show();
44     return 0;
45 }
```

```
This is a test. - length: 15
I like C++. - length: 11
Freeing...
Freeing...
Program ended with exit code: 0
```

Example (Time of a program)

```cpp
#include<iostream>
#include<iomanip>
#include<ctime>
using namespace std;
class timer{
    clock_t start;
public:
    timer();
    ~timer();
};
timer::timer(){
    start = clock();
}
timer::~timer(){
    clock_t end;
    end = clock();
    cout<<setprecision(5);
    cout<< "Elapsed Time: " << (double)(end-start)/(double) CLOCKS_PER_SEC << endl;
}
timer ob;
int main(){
    char c;
    cout<<"Enter a character and press enter."<<endl;
    cin>>c;
    return 0;
}
```

Task

Implement a stopwatch class that emulates a stopwatch that keeps track of elapsed time. Use a constructor to set initial elapsed time 0. Provide two member functions start( ) and stop( ) which will turn on and off the timer respectively.Keep a member function named show( ) that displays elapsed Time. Also have a destructor function which will automatically display elapsed time when timer is destroyed.

Constructor With Parameter

```
#define SIZE 255
class strtype{
    char *p;
    int len;
public:
    strtype(char *ptr);
    ~strtype();
    void show();
};
strtype::strtype(char *ptr){
    p = (char *) malloc(SIZE);
    if(!p){
        cout<<"Allocation Error\n";
        exit(1);
    }
    if(strlen(ptr) >= SIZE){
        cout<<"String is too big"<<endl;
        return;
    }
    strcpy(p, ptr);
    len = (int)strlen(p);
}
strtype::~strtype(){
    cout<<"Freeing..."<<endl;
    free(p);
}
void strtype::show(){
    cout<< p << " - length: " << len <<endl;
}
int main(){
    strtype s1("This is a test."),s2("I like C++.");
    s1.show();
    s2.show();
    return 0;
}
```

```
stack::stack(char c){
    name = c;
    cout << "Constructing stack: " <<name<<endl;
    tos = 0;
}
```

Object Pointer

```cpp
#include<iostream>
using namespace std;
class myClass{
    int a;
public:
    myClass(int x){
        a = x;
    }
    void show(){
        cout<< "a = "<< a << endl;
    }
};
int main(){
    myClass a(10);
    myClass *p;
    p = &a;
    a.show();
    p->show();
}
```

```
a = 10
a = 10
Program ended with exit code: 0
```

Assigning Objects

```cpp
#include<iostream>
using namespace std;
class myClass{
    int a;
public:
    myClass(int x){
        cout<<"Constructor is called with: "<<x<<endl;
        a = x;
    }
    ~myClass(){
        cout<<"Destructing."<<endl;
    }
    void show(){
        cout<< "a = "<< a << endl;
    }
};
int main(){
    myClass a(10);
    myClass b = a;
    a.show();
    b.show();
}
```

```
Constructor is called with: 10
a = 10
a = 10
Destructing.
Destructing.
Program ended with exit code: 0
```

Assigning Objects

```
34  int main(){
35      stack s1('a'),s2('x');
36      s1.push('a');
37      s2.push('x');
38      s1.push('b');
39      s2.push('y');
40      s1.push('c');
41      s2.push('z');
42      s2 = s1;
43      for(int i = 0;i < 3;i++) cout<<"Pop s1: " << s1.pop() <<endl;
44      for(int i = 0;i < 3;i++) cout<<"Pop s2: " << s2.pop() <<endl;
45      return 0;
46  }
47
```

```
Constructing stack: a
Constructing stack: x
Pop s1: c
Pop s1: b
Pop s1: a
Pop s2: c
Pop s2: b
Pop s2: a
Program ended with exit code: 0
```

Error: Assigning Objects

```
26 strtype::~strtype(){
27     cout<<"Freeing..."<<endl;
28     free(p);                                   Thread 1: signal SIGABRT
29 }
30 void strtype::show(){
31     cout<< p << " - length: " << len <<endl;
32 }
33 int main(){
34     strtype s1("This is a test."),s2("I like C++.");   2 ⚠ ISO C++11 does not allow conversion from string literal to 'char *'
35     s1 = s2;
36     s1.show();
37     s2.show();
38     return 0;
39 }
```

Closer look into classes 〉 Thread 1 〉 6 strtype::~strtype() Line: 37 Col: 15

```
I like C++. - length: 11
I like C++. - length: 11
Freeing...
Freeing...
Closer look into classes(4347,0x1edee9b40) malloc: *** error for object 0x600003e00100: pointer being freed was not
    allocated
Closer look into classes(4347,0x1edee9b40) malloc: *** set a breakpoint in malloc_error_break to debug
(lldb)
```

Solution Copy Constructor

Object Parameter (PassByValue)

```cpp
#include<iostream>
using namespace std;
class Integer{
    int number;
public:
    int get_number(){
        return number;
    }
    void set_number(int a){
        number = a;
    }
};
void square_it(Integer a){
    a.set_number(a.get_number()*a.get_number());
    cout<<"The number in copy of a is: "<<a.get_number()<<endl;
}
int main(){
    Integer a;
    a.set_number(10);
    square_it(a);
    cout<<"The number in a is: "<< a.get_number() << endl;
}
```

```
The number in copy of a is: 100
The number in a is: 10
Program ended with exit code: 0
```

Object Parameter (PassByAdress)

```cpp
#include<iostream>
using namespace std;
class Integer{
    int number;
public:
    int get_number(){
        return number;
    }
    void set_number(int a){
        number = a;
    }
};
void square_it(Integer *a){
    a->set_number(a->get_number()*a->get_number());
    cout<<"The number in copy of a is: "<<a->get_number()<<endl;
}
int main(){
    Integer a;
    a.set_number(10);
    square_it(&a);
    cout<<"The number in a is: "<< a.get_number() << endl;
}
```

```
The number in copy of a is: 100
The number in a is: 100
Program ended with exit code: 0
```

Object Parameter (ConstructorCall)

```cpp
#include<iostream>
using namespace std;
class Integer{
    int number;
public:
    Integer(){
        cout<< "Constructor is called." <<endl;
    }
    ~Integer(){
        cout<<"Destructor is called." <<endl;
    }
    int get_number(){
        return number;
    }
    void set_number(int a){
        number = a;
    }
};
void square_it(Integer a){
    a.set_number(a.get_number()*a.get_number());
}
int main(){
    Integer a;
    a.set_number(10);
    square_it(a);
}
```

```
Constructor is called.
Destructor is called.
Destructor is called.
Program ended with exit code: 0
```

Returning an Object

```
3  class Integer{
4      int number;
5  public:
6      Integer(){
7          cout<< "Constructor is called." <<endl;
8      }
9      ~Integer(){
10         cout<<"Destructor is called." <<endl;
11     }
12     int get_number(){
13         return number;
14     }
15     void set_number(int a){
16         number = a;
17     }
18 };
19 Integer square_it(Integer a){
20     a.set_number(a.get_number()*a.get_number());
21     cout<<"A in copy is: " << a.get_number()<<endl;
22     return a;
23 }
24 int main(){
25     Integer a;
26     a.set_number(10);
27     a = square_it(a);
28     cout<<"A is : " << a.get_number() <<endl;
29 }
```

```
Constructor is called.
A in copy is: 100
Destructor is called.
Destructor is called.
A is : 100
Destructor is called.
Program ended with exit code: 0
```

Friend Function

```cpp
#include<iostream>
using namespace std;
class Pair{
    int a,b;
public:
    Pair(int x,int y){
        a = x;
        b = y;
    }
    friend bool is_divisor(Pair mypair);
};
bool is_divisor(Pair mypair){
    if(mypair.a % mypair.b == 0) return true;
    else if(mypair.b % mypair.a == 0) return true;
    return false;
}
int main(){
    Pair a(10,20);
    Pair b(2,7);
    cout<< is_divisor(a) <<endl;
    cout<< is_divisor(b) <<endl;
}
```

```
1
0
Program ended with exit code: 0
```

Friend Function

```cpp
#include<iostream>
using namespace std;
class Car;
class Truck;
class Car{
    int speed;
public:
    Car(int s){
        speed = s;
    }
    friend bool car_has_greater_speed(Car c,Truck t);
};
class Truck{
    int speed;
public:
    Truck(int s){
        speed = s;
    }
    friend bool car_has_greater_speed(Car c,Truck t);
};
bool car_has_greater_speed(Car c,Truck t){
    if(c.speed > t.speed) return true;
    else return false;
}
int main(){
    Car c(20);
    Truck t(10);
    cout<<car_has_greater_speed(c, t)<<endl;
}
```

```
1
Program ended with exit code: 0
```

Friend Function

```cpp
#include<iostream>
using namespace std;
class Car;
class Truck;
class Car{
    int speed;
public:
    Car(int s){
        speed = s;
    }
    bool car_has_greater_speed(Car c,Truck t);
};
class Truck{
    int speed;
public:
    Truck(int s){
        speed = s;
    }
    friend bool Car::car_has_greater_speed(Car c,Truck t);
};
bool Car::car_has_greater_speed(Car c,Truck t){
    if(c.speed > t.speed) return true;
    else return false;
}
int main(){
    Car c(20);
    Truck t(10);
    cout<<c.car_has_greater_speed(c, t)<<endl;
}
```

```
1
Program ended with exit code: 0
```