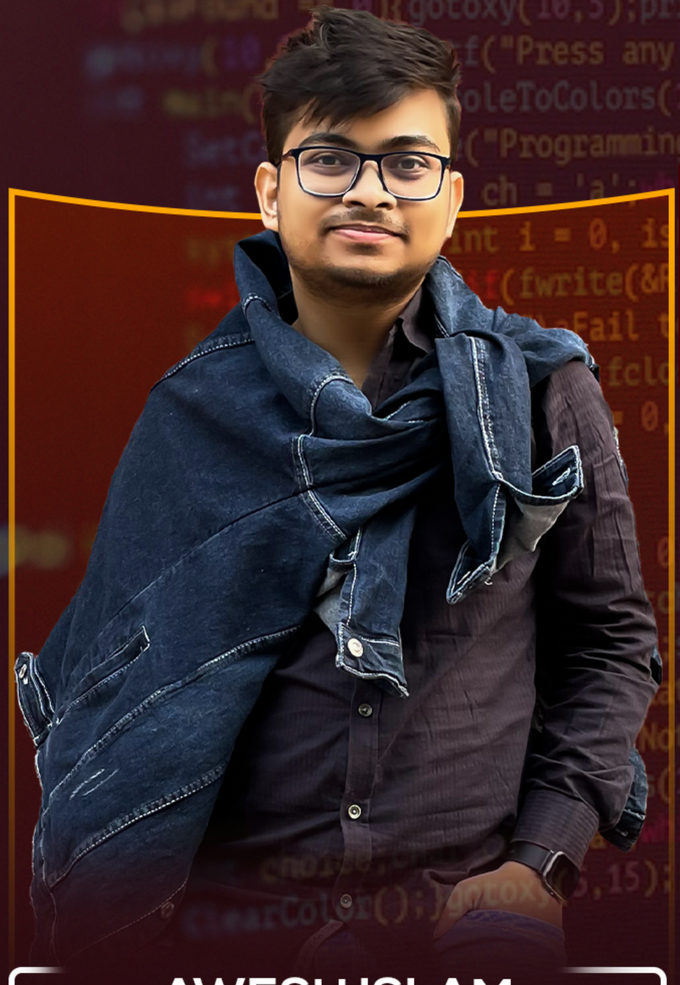
BATCH

RECURSION

C AND C++

PROGRAMMING MASTERCLASS



AWESH ISLAM
BUET, CSE

C++ Class-08

SHAROARE HOSAN EMON
BME , BUET

আমাদের সবগুলো ক্লাস দেখার জন্য ভিজিট করো

https://www.hsccrackers.com/

SCAN ME

Template and Exception Handling

Generic Functions

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

template <class X>
void swap_val(X &a,X &b){
    X temp = a;
    a = b;
    b = temp;
}
int main(int argc, const char * argv[]) {
    int a = 10,b = 20;
    double x = 10.2,y = 10.5;
    cout<<"Before swapping a = "<<a<<" b = "<<b<<endl;
    cout<<"Before swapping x = "<<x<<" y = "<<y<<endl;
    swap_val(a,b);
    swap_val(x,y);
    cout<<"After swapping a = "<<a<<" b = "<<b<<endl;
    cout<<"After swapping x = "<<x<<" y = "<<y<<endl;
    return 0;
}
```

```
Before swapping a = 10 b = 20
Before swapping x = 10.2 y = 10.5
After swapping a = 20 b = 10
After swapping x = 10.5 y = 10.2
Program ended with exit code: 0
```

Generic Functions

```
 8  #include <iostream>
 9  using namespace std;
10
11  template <class X,class Y>
12  void outdata(X a,Y b){
13      cout<<"A is : "<<a<<" B is : "<<b<<endl;
14  }
15  int main(int argc, const char * argv[]) {
16      outdata(10, "hi");
17      outdata<double, int>(10.5, 100);
18  }
19
```

```
A is : 10 B is : hi
A is : 10.5 B is : 100
Program ended with exit code: 0
```

What happens if we overload

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

template <class X,class Y>
void outdata(X a,Y b){
    cout<<"A is : "<<a<<" B is : "<<b<<endl;
}
void outdata(int a,int b){
    cout<<"This is overloaded outdata"<<endl;
    cout<<"A is : "<<a<<" B is : "<<b<<endl;
}
int main(int argc, const char * argv[]) {
    outdata(10, "hi");
    outdata<double, int>(10.5, 100);
    outdata(10, 20);
}
```

```
A is : 10 B is : hi
A is : 10.5 B is : 100
This is overloaded outdata
A is : 10 B is : 20
Program ended with exit code: 0
```

Generic Classes

```
8  #include <iostream>
9  using namespace std;
10
11 template <class data_t> class list {
12     data_t data;
13     list *next;
14 public:
15     list(data_t d);
16     void add(list *node){
17         node->next = this;
18         next = 0;
19     }
20     list *get_next() { return next; }
21     data_t get_data() { return data; }
22 };
23 template <class data_t> list<data_t>::list(data_t d){
24     data = d;
25     next = 0;
26 }
27 int main(){
28     list<char> start('a');
29     list<char> *p,*last;
30     last = &start;
31     for(int i = 1;i < 26;i++){
32         p = new list<char> ('a' + i);
33         p->add(last);
34         last = p;
35     }
36     p = &start;
37     while (p) {
38         cout<<p->get_data()<<endl;
39         p = p->get_next();
40     }
41     return 0;
42 }
```

Generic Classes

```
8  #include <iostream>
9  using namespace std;
10 #define SIZE 10
11 template <class StackType> class stack{
12     StackType stck[SIZE];
13     int tos;
14 public:
15     void init() { tos = 0; }
16     void push(StackType ch);
17     StackType pop();
18 };
19 template <class StackType>
20 void stack<StackType>::push(StackType ch){
21     if(tos == SIZE){
22         cout<<"Stack is Full"<<endl;
23         return;
24     }
25     stck[tos] = ch;
26     tos++;
27 }
28 template <class StackType>
29 StackType stack<StackType>::pop(){
30     if(tos == 0){
31         cout<<" Stack is empty \n";
32         return 0;;
33     }
34     tos--;
35     return stck[tos];
36 }
```

```
37 int main(){
38     stack<char> s1;
39     s1.init();
40     s1.push('a');
41     s1.push('b');
42     s1.push('c');
43     stack<double> d1;
44     d1.init();
45     d1.push(10.2);
46     d1.push(20.5);
47     d1.push(6.6);
48     for(int i = 0;i < 3;i++){
49         cout<<s1.pop()<<endl;
50     }
51     for(int i = 0;i < 3;i++){
52         cout<<d1.pop()<<endl;
53     }
54 }
55
```

```
c
b
a
6.6
20.5
10.2
Program ended with exit code: 0
```

Generic Class with two datatypes

```cpp
#include <iostream>
using namespace std;
#define SIZE 10
template <class Type1,class Type2> class myClass
{
    Type1 a;
    Type2 b;
public:
    myClass(Type1 a,Type2 b){
        this->a = a;
        this->b = b;
    }
    void show() {cout<<a<<" "<<b<<"\n";}
};
int main(){
    myClass<int,double> ob1(10,100.10);
    myClass<char, string> ob2('x',"Hello World");
    ob1.show();
    ob2.show();
}
```

```
10 100.1
x Hello World
Program ended with exit code: 0
```

Exception Handling

```cpp
#include <iostream>
using namespace std;
int main(){
    cout<<"Start\n";
    try{
        cout<<"Inside a try block..."<<endl;
        throw 10;
        cout<<"This will not be executed"<<endl;
    }
    catch(int i){
        cout<<"Caught a number..."<<endl;
        cout<<"The number is: "<<i<<endl;
    }
}
```

```
Start
Inside a try block...
Caught a number...
The number is: 10
Program ended with exit code: 0
```

What happens if no matching catch

```cpp
#include <iostream>
using namespace std;
int main(){
    cout<<"Start\n";
    try{
        cout<<"Inside a try block..."<<endl;
        throw 10;
        cout<<"This will not be executed"<<endl;
    }
    catch(double i){
        cout<<"Caught a number..."<<endl;
        cout<<"The number is: "<<i<<endl;
    }
}
```

Template 〉 Thread 1 〉 9 main

```
Start
Inside a try block...
libc++abi: terminating due to uncaught exception of type int
(lldb)
```

Exception Handling from a function

```cpp
#include <iostream>
using namespace std;
void Xtest(int test){
    cout<<"Inside Xtest, test is: "<<test<<endl;
    if(test) throw test;
}
int main(){
    cout<<"Start\n";
    try{
        cout<<"Inside a try block..."<<endl;
        Xtest(0);
        Xtest(1);
        Xtest(2);
    }
    catch(int i){
        cout<<"Caught a number..."<<endl;
        cout<<"The number is: "<<i<<endl;
    }
}
```

```
Start
Inside a try block...
Inside Xtest, test is: 0
Inside Xtest, test is: 1
Caught a number...
The number is: 1
Program ended with exit code: 0
```

Exception Handling from a function

```cpp
#include <iostream>
using namespace std;
void Xtest(int test){
    cout<<"Inside Xtest, test is: "<<test<<endl;
    try{
        if(test) throw test;
    }
    catch(int i){
        cout<<"Caught a number..."<<endl;
        cout<<"The number is: "<<i<<endl;
    }
}
int main(){
    cout<<"Start\n";
    Xtest(0);
    Xtest(1);
    Xtest(2);
}
```

```
Start
Inside Xtest, test is: 0
Inside Xtest, test is: 1
Caught a number...
The number is: 1
Inside Xtest, test is: 2
Caught a number...
The number is: 2
Program ended with exit code: 0
```

Exception Handling

```cpp
#include <iostream>
using namespace std;
void Xtest(int test){
    cout<<"Inside Xtest, test is: "<<test<<endl;
    try{
        if(test == 0) throw 11;
        if(test == 1) throw 123.3;
        if(test == 2) throw  "hello";
    }
    catch(...){
        cout<<"Caught an exception..."<<endl;
    }
}
int main(){
    cout<<"Start\n";
    Xtest(0);
    Xtest(1);
    Xtest(2);
}
```

```
Start
Inside Xtest, test is: 0
Caught an exception...
Inside Xtest, test is: 1
Caught an exception...
Inside Xtest, test is: 2
Caught an exception...
Program ended with exit code: 0
```

Re-throw

```cpp
#include <iostream>
using namespace std;
void Xtest(int test){
    cout<<"Inside Xtest, test is: "<<test<<endl;
    try{
        if(test == 0) throw 11;
        if(test == 1) throw 123.3;
        if(test == 2) throw  "hello";
    }
    catch(...){
        cout<<"Caught an exception..."<<endl;
        throw;
    }
}
int main(){
    cout<<"Start\n";
    try{
        Xtest(0);
        Xtest(1);
        Xtest(2);
    }catch(...){
        cout<<"Inside another catch"<<endl;
    }
}
```

```
Start
Inside Xtest, test is: 0
Caught an exception...
Inside another catch
Program ended with exit code: 0
```

throw with new

```
8  #include <iostream>
9  using namespace std;
10
11 int main(){
12     double *p;
13     do{
14         try{
15             p = new double[1000000];
16         }catch(bad_alloc xa){
17             cout<<"Allocation failure"<<endl;
18         }
19     }while(p);
20 }
```

New no throw

```
8  #include <iostream>
9  using namespace std;
10
11 int main(){
12     double *p;
13     do{
14         p = new(nothrow) double[1000000];
15         if(!p){
16             cout<<"Allocatin error"<<endl;
17         }
18     }while(p);
19 }
```