Exception Handling in C++

Contents:

- Pointers to Object
- Array of Pointer to Object
- Exception Handling
- Exception Handling Model
- Exceptions in Constructors & destructors
- Streams Computation with console
- Stream computations on Files

Pointer to Objects

- Pointer to objects are used when objects are created during execution.
- Similar to variables address operator & can be used to get the address of object.

Syntax:

class_name *ptr_object;

ptr_object = &object;//address of a statically created object

ptr_object = new class_name; //object created dynamically

Programming of Dynamic Objects

```
#include<iostream>
class dyn_obj        {
public: int data1; char data2;
dyn_obj() {
cout<<"constructor";
data1 = 1; data2 = 'a'; }
~dyn_obj() {
cout<<"destroctor"; }
void show(){
cout<<data1<<endl<<data2;
      }
};
```

```
void main()
 {
dyn_obj *ptr_obj;
ptr = new dyn_obj;
ptr -> show();
delete ptr;
}
```

Array of Pointer to Objects

- Used to often handle group of objects.
- Objects need not necessary reside contiguously in memory.

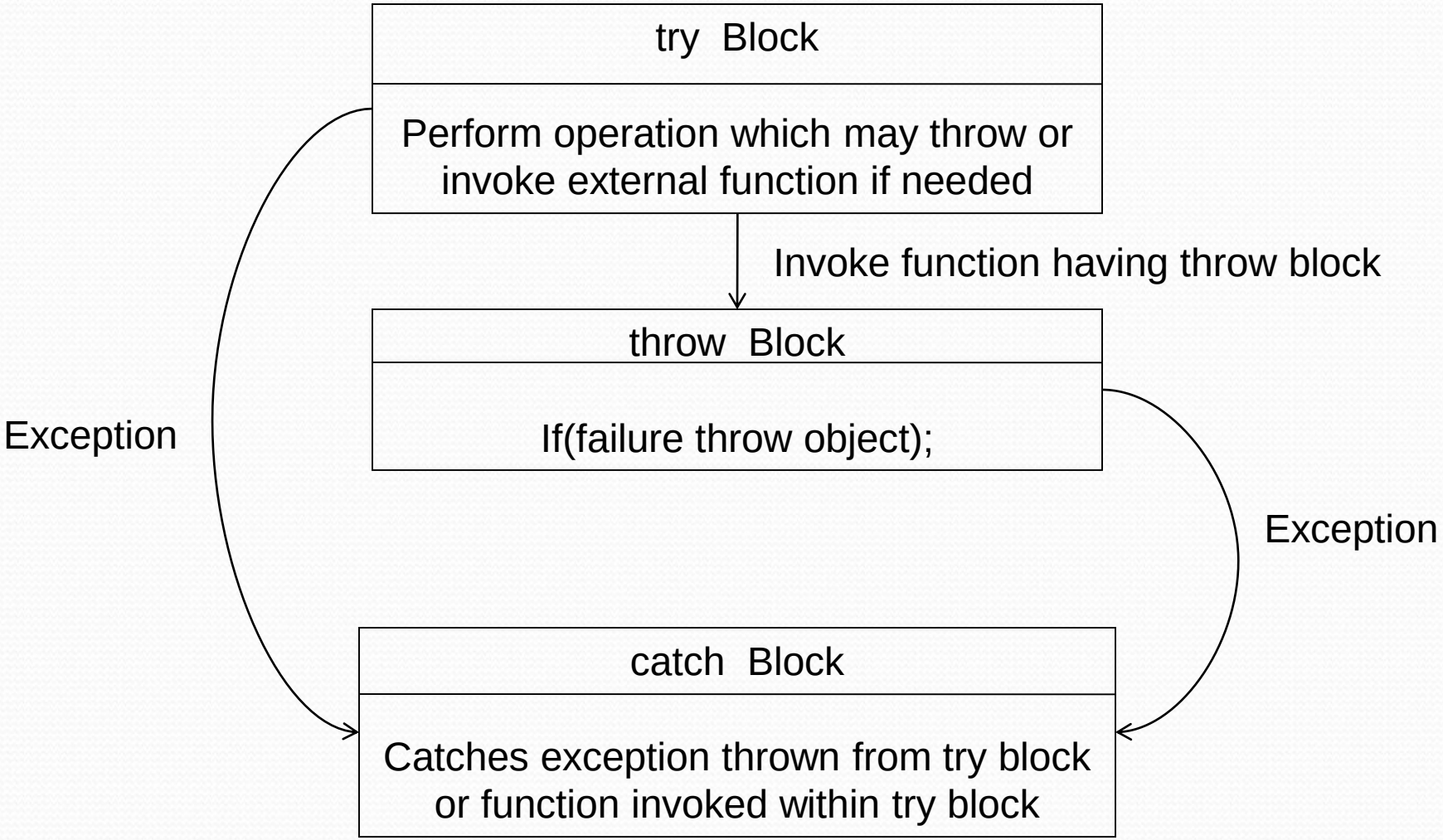
```
#include<iostream>
#include<string>
class student {
private: int r_num;
        char name[20];
public:
void setdata(int r, char *n){
        r_num = r;
        strcpy(name, n);}
};
```

```
void main(){
int i,count=0,rn;
char name[20];
student *s[10];
for(i=0;i<10;i++) {
cin>>rn; cin>>name;
s[i] = new student;
s[i]->setdata(rn, name);
count++; }
}
```

Exception Handling

- Transfers control & information from point of exception in a program to an exception handler.
- Used to support error handling & fault tolerant computing.
- Two ways to handle exceptions are:
  - ✓ With error checks on return value.
  - ✓ setjmp & longjump mechanism.

Exception Handling Model



Exception Handling Constructs

- *try* Construct
  - Defines boundary within which an exception can occur.
  - Block of code in which exception can occur must be prefixed by keyword *try*.
- *throw* Construct
  - Keyword *throw* is used to raise an exception.
  - *thorw* initializes an object of type T to throw.
- *catch* Construct
  - Exception handler is indicated by *catch.*
  - *Used immediately after try block.*
  - *Consecutive catch handler can also occur.*

Exception Handling Program

```
#include<iostream>
class number {
   private: int num;
   public: class DIVIDE {};
      int div(number num2){
          if(num2.num==0)
             throw DIVIDE();
          else
            return (num/num2.num);
          }
   };
```

```
void main() {
   number num1,num2; int r;
   try {
          r=num1.div(num2);
          cout<<"succeeded";
        }
   catch(number:DIVIDE)
        { cout<<"failed";
           cout<<"Exception:
Divide by Zero";
          return 1;
         }
Cout<<r;  }
```

Exceptions in Constructors & destructors

- On exception, the copy constructer is invoked.
- Copy constructor initializes temporary object at the throw point.
- When program flow is interrupted by an exception, destructors are invoked for all objects constructed from entry point of try block.
- If exception is thrown during construction of object, destructor will be called only for fully constructed objects.

Console Stream Classes

- ios class
  - Provides operation common to both input & output
- istream class
  - Derived class of ios
  - Defines get(), getline(), read() functions.
- ostream class
  - Derived class of ios
  - Defines put() & write() functions.
- iostream class
  - Derived from istream & ostream.

Stream computations on Files

The actions performed by classes related to filen management are:

- fstreambuf: Supports operations common to the file stream. Served as base class.
- filebuf : Sets the buffer to read /write.
- ifstream: Supports i/p operations.
  - get(), getline(), seekg(), tellg(), read().
- ofstream: Supports output operations.
  - put(), seekp(), tellp(), write().
- fstream: Supports simultaneous i/p & o/p operations.