9/10/19

Namespace

```
#include<iostream.h>
Using namespace std;
Namespace first
{  void enter()
   {  cout<<" we are in namespace";
   }
}
namespace second
{  void enter()
   {  cout<<" we are in
      namespace second";
   }
}
int main()
{  first :: enter();
   second :: enter();
   getch();
   return 0;
}
```

15/10/19

Page No.
Date

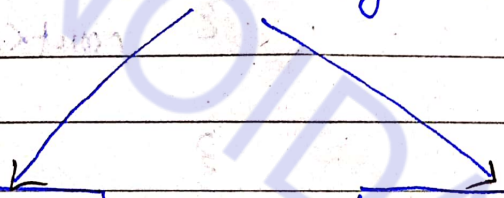
Exception Handling

- Try
- Catch
- Throw

It is a technique of C++ used to handle the unusual and rare situations that occurs in a program.
→ (i.e. no syntax mistake)

They are not errors but they can take the compiler into an incorrect situation
→ (or results)

Exceptions are of 2 types



- Synchronous
- Asynchronous

Ex → new allocat

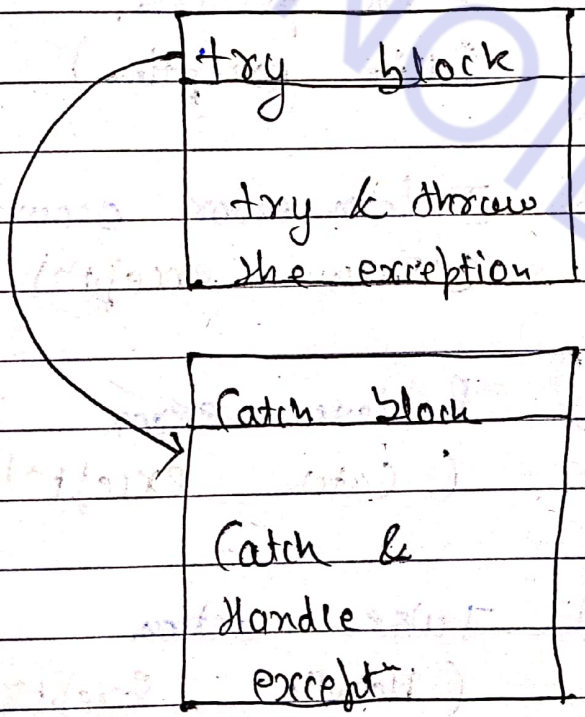
Synchronous are those which takes the program out of range index and overflow.

Ex → divide by 0

Whereas asyncronous exception are those that are caused by events beyond program control

Page No.
Date

Try
{ // try code
Throw
}
Catch
{ // except^n output
}



Throw except^n object

The purpose of this Mechanism is to provide means to detect & report an exceptional circumstance.

It provides an appropriate actn that should be taken to handle situation.

The mechanism provides a saparate error handling code that performs following task.

(A) Find problem
(Hit exception)

(B) Inform error occurred
(Throw exceptn)

(c) Recieve error
(Catch exceptn)

(d) Take action.
(Handle Exceptn)

16/10/19

Page No.
Date

```
try
{
        throw exception;
}

catch (type Arg)
{

}
```

{try}

The code which can throw any exception is enclosed in a try block.

{catch}

It is intended to catch the error and handle the exceptional condition. We can have multiple catch blocks to handle different type exception.

And perform different actions when exception occurs,

{Throw;}

It is used to throw exception to exception handler. It is used to communicate information about unusual situation.

A throw expression accepts one parameter and that parameter is passed to exception handler.

throw statement used when we explicitly want exception to occur.

#.

```
int main()
{   int x = 10, y = 0, z;
    z = x/y;
    return 0;
}
```

try
{

```
#include <iostream.h>

int main ()
{   int x = 10, y = 0, z;
```

Page No.
Date

try
{
if (y==0)
throw "divide by zero";
else
z = x/y;
}
catch (exception e)
{ cout<< "divide by zero";
}
return 0;
}

Re->

try
{ z = x/y;
if (y==0)
throw 'z';
}

Page No.
Date

```
catch ( int z)
{   cout<<" divide by zero";
}
return 0;
}
```

18/10/19

Mam →

```
#
// num div by zero

int main()
{ int a,b;
  cout<<" Enter Value for -
  cin>>a; numerator ";
  cout<<" Enter b ";
  cin>>b;

  try
  {
     if (b!=0)
     {  cout<<" Divide the Numbers"
                <<a/b;
     else
        throw(b);
  }
```

```
catch
{
    cout<<" Exception caught in the
           division";
}
getch();
}
```

Q2 Print 1-20, if above 20 then exception

Sol #

```
int main()
{   int Num;
    cout<<" Enter Num 1 to 20";
    try
    {
        cin>>a
        if (a>=1 && a<=20)
            cout<<a;

        else
            throw (a);
    }
    catch (int a)
    {   cout<<" Value not in
               range 1 to 20";
    }
    getch();
    return 0;
}
```

C -> Calloc & Malloc

Page No.
Date

23/10/19

Dynamic Memory Allocation

(New and Delete)

New Optr :-

It creates memory at run time.

i.e.

while declaring array we must provide size of array to allocate memory at compile time.

If we want to allocate memory at run time we must use new optr followed by data type.

Syntax :-

```
ptr = new data_type [size];
ptr = new data_type;
```

allocate for fixed no. of elements.

Page No.
Date

Delete optr :-

Used to deallocate memory created by New optr at run time.

Once memory is no longer needed it should be freed so that the memory becomes available again for other request of dynamic memory allocation.

Syntax :-

→ delete ptr; // deallocate memory for only 1

→ delete [] ptr; // deallocate for Array

/* WAP to implement New & Delete */

(PTO)

```
#include <iostream.h>
#include <conio.h>

void main()
{
    int *arr[10]; // *arr
    int size;
    cout<<"Enter size: ";

    cin>>size;

    arr = New int[size];

    cout<<"Enter elements ";

    for(int i=0; i<size; ++i)

        cin>>arr[i]; //(arr+i)

    cout<<"print : ";

    for(int j=0; j<size; ++j)

        cout<<arr[j]; *(arr+i)
    delet[size] arr; //check [] arr
    getch();
}
```

Page No.
Date

=> Record files handling

- Fopen
- Fclose
- EOF

```
/*  #   <iostream.h>
        <conio.h>
        <fstream.h>

    int main()
    {   ofstream file

        file.open("bca.txt");
        file.close();
        cout<<"file has been created";

        getch();
        return 0;
    }
```

Page No.
Date

★ Fseek : It is a function of fostream library that allows us to seek an arbitrary position in a file. It is mainly used in file handling the position of the character to be extracted from a file.

We have 2 types of seek funct^n

- seekg ( )
- seekp ( )

seekg ( ) used to move the get pointer to a desired locat^n with respect to a reference point.

seekp ( ) used to move the put pointer to a desired locat^n with respect to a refrence pointer.

Syntax

file_pointer.seekg (no. of bytes, refrence

file_pointer.seekp (bytes, refrence

Page No.
Date

★ Rewind :- Rewinds to set the file position indicator for the stream pointed to by stream to the begning of the file.

Rewind() takes the pointer and set the file position to the begining of the file of the given stream.

Syntax :- `Void rewind (file* stream)`

Stream is the pointer to file object that identifies the stream.

This function does not return any value.

★ Getc() :- This function of stream class reads the next character from a file and it takes a file pointer to the file so that a file can be read.

★ putc() :-

bca = fopen("ims.txt");

Page No.
Date

putc() of file handling is used to display on std. output or it enable to write in a file after which getc is able to read the character from a file.

06/11/19

open(), close()

Stream classes

1) ifstream → Read Only.
2) ofstream → Write + Create
3) fstream → Read + Write

Create + Write + close

```
#include <iostream.h>
#include <fstream.h>
int main()
{
    ofstream bca;
    bca.open("ims.txt");
```

object

bca << "I am writing in a file";
bca << "This is another line";
cout << "I am writing in a file";
bca.close();
return 0;
}

* * for counting characters and space.

#
#

void main()
{

```
int main()
{
    is_open();
    bca.open("im.txt");
    if(bca.is_open())
    {
    bca<<"
    bca<<"
    }
    else
    {cout<<" doesn't exist",
    }
    bca.close();
}
```