21.11.2022
Monday

→ exact same fn from base we can differentiate them.

Function Overriding.

VIRTUAL FUNCTION

① Essential requirement of Polymorphism is its ability to refer to object without any regard to their classes.
② We use the ptr. to the Base Class to refer to all the DC
③ The compiler ignores the contents of the pointers and chooses the member fn that matches the type of pointer.
④ So, Polymorphism is achieved using a virtual fn
⑤ When we use same fn name in both BC & DC the fn in BC is declared as virtual using keyword "VIRTUAL" preceding its normal declaration.
⑥ When fn made virtual. C++ determines which fn to use at runtime based on the type of object pointed to by the Base pointer rather than type of the pointer.

```
class B {
    public:
        void display () { cout << "Base display "; }
        virtual void show () { cout << "Base show "; }
};
class D: class B {
    public:
        void display () { cout << "derived display "; }
        void show () { cout << "derived show "; }
};
int main () {
        B B1;
        D D1;
        B *ptr;
    cout << " ptr points to base ";
```

Scanned with CamScanner

Runtime Polymorphism

↳ Runtime के दौरान Ptr के content से पता चलेगा which f^n would run?

```
ptr = &B1;
ptr -> display();      // Base display
ptr -> show();         // Base show
cout<<"ptr points to derived";
ptr -> &D1;
ptr -> display();      // ~~derived~~ Base display   [Type of ptr देखेगा]
ptr -> show();         // Derived show              [content देखेगा]
}
```

Virtual f^n in BC

same name होना जरूरी

O/P:
Base display
Base show
Base display
Derived show

Pure Virtual Function:

→ virtual f^n of BC DC में होना ही चाहिये

① Pure Virtual function (PVF) declared in a BC that has no defination relative to BC. ~~It~~

② It is a "do nothing function".

```
virtual void display() = 0;
```
// supporting DC

③ class containing pure virtual f^n can't be used to declare any objects of its own. → ~~Abs~~

Such classes are called as Abstract Base Class

↳ used for Inheritance

At least one pure virtual f^n

Rules for VF: ① can't be static
② ~~Accessed by obj. ptrs~~
③ VF can be friend of another class
④ A VF in BC might not be used
⑤ VF in BC, there is no necessity to define in DC

Scanned with CamScanner

23-11-2022
Tuesday

new
delete } dynamic memory allocation

Date / / Page no.

Virtual Destructor

(Specifically no one not but can be asked)

① A problem can occur when using polymorphism to process dynamically allocated objects to a class heirarchy.

```
Base *ptr = New derived;
:
delete ptr;
```

② If a derived class object with a non virtual destructor is destroyed by applying delete operator to a base class ptr. to the object the C++ standard specifies that the behaviour undefined.

③ Here the destructor of the base class will be called but not the one of DC

④ Sol^n of this scenario is "Virtual Destructor in the BC"

```
virtual ~Base() {
}
```

⑤ Now if the object in the heirarchy is destroyed explicitly by applying the "delete operator" to the BC pointer. The destructor for the appropriate class is called based on the object to which the BC pointer points.

⑥ BC destructor automatically executes after the DC destructor

Scanned with CamScanner

Date: / / Page no:

→ C++ ✓ C ✗ but can be done with the help of if else

Exception Handling (EH)

(1) One of the advantage of C++ over C is EH

(2) Exceptions are runtime anomalies / abnormal cond^n that a program encounters during its execution.

(3) A C++ exception is a response to an exceptional circumstances that arise ~~by the~~ while a program is running such as an attempt to divide by 0.

(4) Exceptions provides a way to transfer control from one part of a program to another.

(5) C++ exception handling is build upon ⊕ 3 keywords.

"try", "catch" and "throw"

try & catch are compulsory

exception occurs ② ① if exception = true

prt1 → prt2

try catch

throw can occur alone

or try → catch

~~catch~~ → catch

→ catch.

Throw: A program throw an exception when a problem shows up. This is done using a throw keyword.

Catch: A program catches an exception with an exception handler at the place in a program where you want to handle the problem.

The catch keyword indicates, the catching of an exception.

Try: Try block identifies a block of code for which a particular exception will be activated. It followed by 1 or more catch block.

File handling → File does not opens up (Runtime error)

Scanned with CamScanner

24.11.2022
Tuesday

Date __/__/__ Page no. ____

6) The main purpose of Exception Handling programm is to detect and manage runtime errors

7) Whenever a caller of a f^n detects an error without exception handling then its is very difficult to trace, check & handle in a complex and big software.

Ex: divide by 0, memory not allocated.
memory out of bound (full)
file not open.
] Runtime

```
try {
    ===                    // checking code
    throw (arguments)
    ===
}
catch (arguments) {
    ===                    // message
}
```

together

```
class sample
{ private:
      float a, b;
  public:
      void getdata() {
          cin >>a>> b;
      }
      void divide () {      int n, float f;
          String str;
          try {
              if (b==0)                  else if (b==1) throw n;
                  throw str;             else if (b==2)
                                             throw f;
              else {
```

Scanned with CamScanner

Types of Exception Handling

Date: / / Page no:

```
float temp = a/b;
    cout << temp;
  }
}
catch (String str){
    cout << "Exception"; }
}
int main() {
    sample obj.;
    obj.getdata()
    obj.divide();
{
```

```
~~catch~~ catch(int n) {
    cout<<" Integer Type Error"
    <<endl
catch (float f) {
    cout<<" float type"
catch(...)
```

If catch is not written the exception will get other Runtime.

catch(...) → generalised catch can catch any through type

↳ Handle/catch all ~~type~~ of exceptions

NESTING OF TRY BLOCKS

Try के अंदर Try

When Try blocks are nested and a throw occurs in a function called by an inner try ~~block~~ block. Control is transferred downward through the nested try Blocks until the ~~next try block~~ first catch block is found whose argument matches the argument of throw. ~~throwing~~ exception

Scanned with CamScanner

Date ___/___/___ Page no. _____

```
try {
    func1 ();    // can deal only type error / only specific error
    try {
        func2 ();    // can deal type error
    }
    catch (specific error) { /* ... */ }
    func3 ();    // only type error
}
catch (type Error) { /* ... */ }
```

(never come here)

IMP (i) In this example if specific error is thrown within the inner try block. The exception is caught by inner catch block. and func3 is called

(ii) If specific error is thrown after the inner Try block it is not caught and program terminates.

(iii) If Exception thrown from func2 in the inner try block is type error the program skips out of both ~~wks~~. ~~to 2~~ to 2nd catch block without running f^n 3.

Scanned with CamScanner

28-11-2022
Monday

← Parallel programming
OpenCL
Kudar
(NVIDIA)
No. of CPUs
GPGPU
GPU
CPU

Date: / / Page no:

Process → Program in execution
↳ sequence of raw data (can be secondary memory)
↓ execute
↓
Primary memory
↓
RAM

↓
Hard Disc / SSD

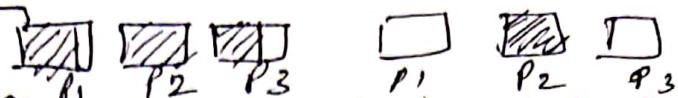
Why such?
cost effective [Price (RAM) > Price (SSD)]
and space (RAM) < space (SSD)]

OS
(Operating System)
↳ Interface betn user & Hardware

Process :- (i) Process means any program is in execution.
(ii) OS maintains process control ~~logs~~ blocks.
(iii) Process control blocks control the information of any process.
↳ (PCB) → Program Id / Process Id (OS)
→ Priority
(iii) PCB contains the information about process ~~for~~
for example: Process priority, Process Id, Process state.
(iv) A process can create other processes which are known as child processes
(v) Processes are isolated means they do not share memory ~~which~~ with any other processes (Independent Tasks)
(vi) Even with single core processor due to OS we can run multiple processes simultaneously.
(vii) Different processes get the processor according to the scheduling ~~done~~ is defined by the OS.
(Time Sharing System)
(viii) Processes get content switch and other processes gets overhead 65 ms work → 25 ms Block.

Scanned with CamScanner

30-11-2022 Wednesday



CPU if a ~~cpu~~ process not get completed in its assigned time.

In this way, multiple processes share a single CPU.

Thread :- (i) A thread is a ~~path~~ path of execution within a process

(ii) A process can contain multiple threads.

(iii) A thread is also known as lightweight Process.

(iv) By this we have an idea to achieve concurrency by dividing a process into multiple thread.

Webbrowser (Process)

Tab1 Tab2 Tab3 (Threads)

google.com Yahoo.com

(v) The primary difference betn process & thread is that thread within the same process run in a shared memory span, while processes run in separate memory span.

(vi) Threads are not independent of one another like ~~one another~~ process are & as a result thread share with other thread their code section, data section & OS resources but like process a thread has its own program counter register sets & stack space.

Concurrency : (i) Concurrency means multiple computation are happening at the same time.

(ii) Concurrency is everywhere in modern program

ex- Multiple application run on a computer

Multiple computer in a network

Multiple processors in a computer [each processor each task]

(iii) Concurrency is ~~so~~ essential in modern programming.

Scanned with CamScanner

Date __/__/__ Page no: _____

Ex. Website must handle multiple simultaneous user

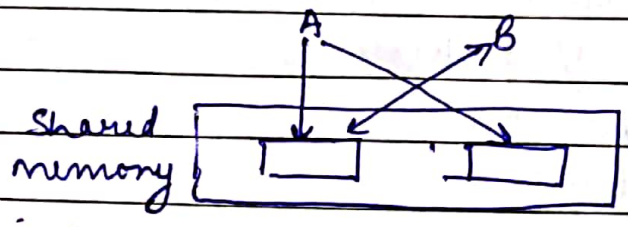
Models for concurrent Programming:

There are 2 Models:

(1) Shared Memory (2) Message Passing

(1) **Shared Memory:** In the shared memory model of concurrency, concurrent modules interacts reading & writing shared objects in memory.

Example

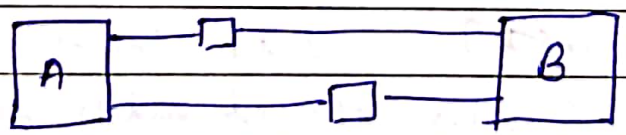


A and B might be 2 processors in the same computer sharing the same Physical memory

A and B might be programmed running on same computer sharing a common file system with file they can read & write.

A and B might be 2 threads in the same program.

(2) **Message Passing:** In the message passing model concurrent modules interact by sending message to each other through a communication channel.



A & B might be 2 computers in a network (nothing shared) but can interact with each other on Network

A & B can be web browsers and Web servers.

Scanned with CamScanner

Date: / / Page no:

Multithreading in C++

(i) Multithreading support was introduced in C++.

(ii) Thread Classes and related functions are defined in the thread header file.

(iii) To start thread we simply need to create a new thread object and pass the executing code to be called into the constructor of the object.

(iv) Once the object is created a new thread is launched which will execute the code specified in ~~in~~ callable.

A Callable can be of 3 type

(a) function pointer

(b) function object

(c) Lambda function (without name fn)

```
# include <iostream>
# include <thread>
using namespace std;
void func() {
    cout << "Thread";
}

int main() {
//    func();
// }
    thread t1 (func)        // t1 start running
    t1.Join();              // main thread waits for t1 to finish
}
```

Normal

→ If t1 not ends & program goes to end of main() ∴ put it before end of main()

Scanned with CamScanner

Date. / / Page no.

```
~~int~~ #include <iostream>
#include <thread>
using namespace std;
void func(int x, int y){
    cout << " Thread " << x << y;
}
int main(){
    int y=10; int z=5;
    thread t1(func,y,z)      // t1 start running
    t1.Join(); // main thread waits for t1 to finish
    t2.Join();
}
```

~~thread~~ t2(func,z,y)

reduces time

Q) for printing even no. from 1 to 1 Lakh — 1,00,000 — t1

" odd " . — t2

Ans:

```
#include <iostream>
#include <thread>
using namespace std;
void evenN(){
    for(int i=1; i<=100000; i++){
        if(i%2==0) cout<<i<<" "; }
}
void addN(){
    for(int i=1; i<=100000; i++){
        if(i%2!=0) cout<<i<<" "; }
}
```

Scanned with CamScanner

Date: / / Page no:

```
int main(){
    thread t1 (evenN);
    thread t2 (oddN);
    t1.join();
    t2.join();
    return 0;
}
```

Scanned with CamScanner