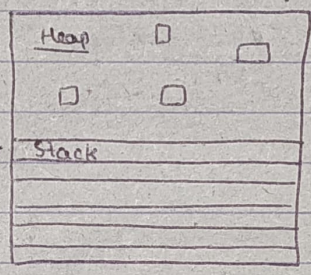
Date: / /
Page:

Day - 67

Memory Management

⇒ int n;
cin >> n;
int arr[n]; X (This is not allowed)
vector<int> v(n); ✓

⇒ We have two ways to store data —
Stack & Heap.



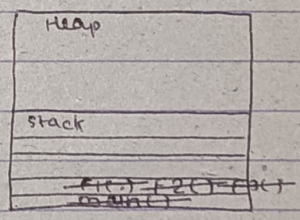
Dynamic Memory ← 4GB

Static Memory ↙

stack
→ Always have low memory
→ 8MB

void F1(){ } void F2(){ } void F3(){ }

⇒ int main(){
int x, y, z;
x=10, y=20, z=30;
F1(); { }
F2();
F3();
}



⇒ when F1() completes their work then it will remove from stack.

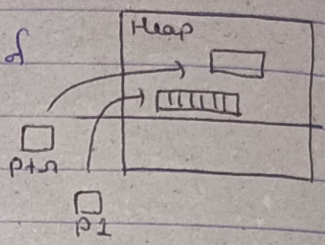
⇒ Same goes to main().

⇒ int arr[n]; → This is not allowed because if the size of n is greater than stack memory then problem or error will come.

Date: / /
Page:

⇒ If we want to store our variables in heap then we have to use a keyword — 'new'

⇒ new int ;
This will return address of that region.



⇒ int *ptr = new int ;
float *ptr1 = new float ;

For array
int *p1 = new int [20]

⇒ Here, the created pointer will store in stack because then we can access the address of that pointers. will

⇒ But if pointers store in heap then we can't access it.

⇒ After our main() completes, stack will automatically releases memory but heap will not do this automatically.

⇒ So, to releases memory from heap, we have to use 'delete' keyword.

Ex: delete ptr;
delete ptr1;
delete [] p; → for array

⇒ Vector stores in heap and after completing the work vector automatically releases from memory

⇒ Dynamic Memory Allocation: when we allocate memory at run time.

Date : / /
Page:

⇒ Static Memory Allocation: when we allocate memory at compile time.