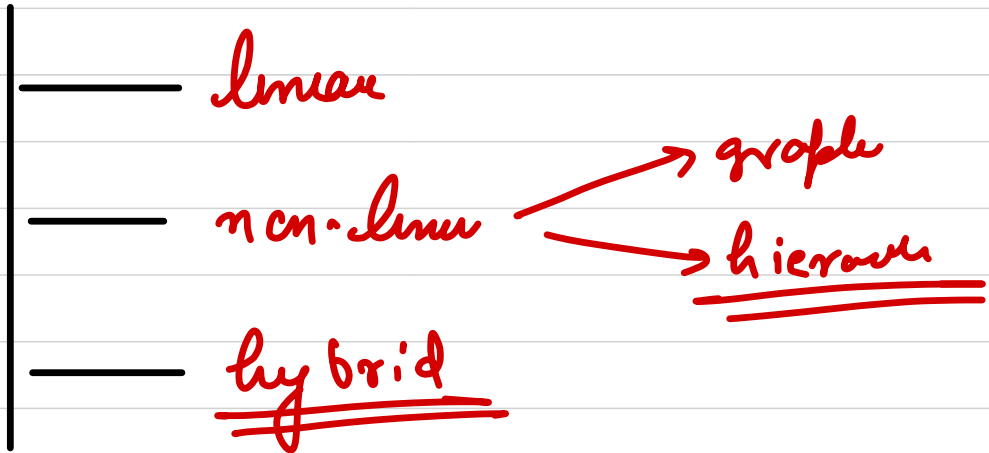


Data Structures

C++

these are storage mechanisms for storing data in different orientation based on use cases.



Arrays

↳ Arrays are linear data structure which store homogenous data in a sequential order in consecutive memory locations.

and is of fixed Size.



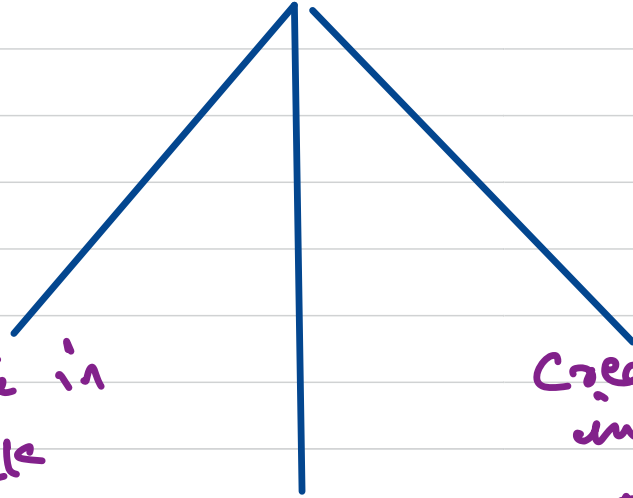
← in memory



consecutive memory block

How to create arrays

Memory



Create in
Stack

<type> <name>[size]

Create a
global

Create
in heap

type* <name> = new int[5]

size
↑

new int;

Code
program

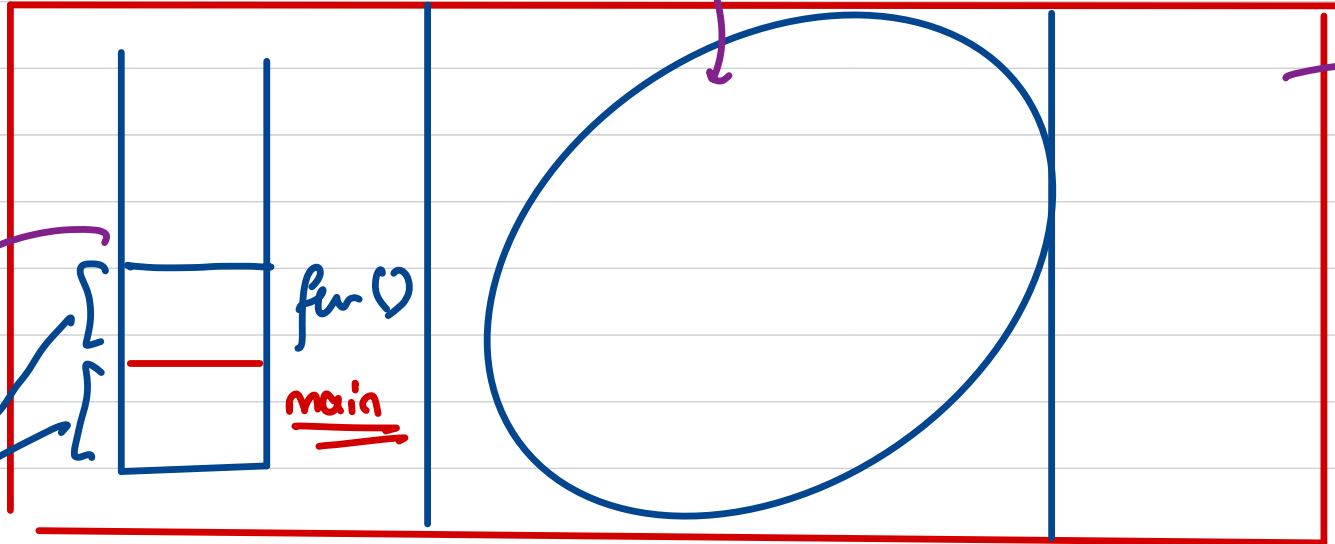
run

process

(loaded in the RAM)

It consumes some
memory of RAM

Big Pool of
memory → heap



Call stack ^{→ LIFO} → It is a part of memory which

keeps track of function calls. Whenever we call a functⁿ, we add an element called as stack frame in our call stack. In the frame we store all local variables of the function call.

```

1  #include<iostream>
2
3  void gun() {
4      std::cout<<"0000 this is a gun\n";
5  }
6
7  void fun() {
8      gun();
9      std::cout<<"Hey this was fun\n";
10 }
11
12 int main(int argc, char const *argv[])
13 {
14     int x = 10;
15     → fun();
16     std::cout<<x<<"\n";
17     → return 0;
18 }
19

```

Compiler

000 this is a gun
hey this was fun
10

first funcⁿ to be executed
is main.

0

call stack



→ Situations in which stack frame is removed

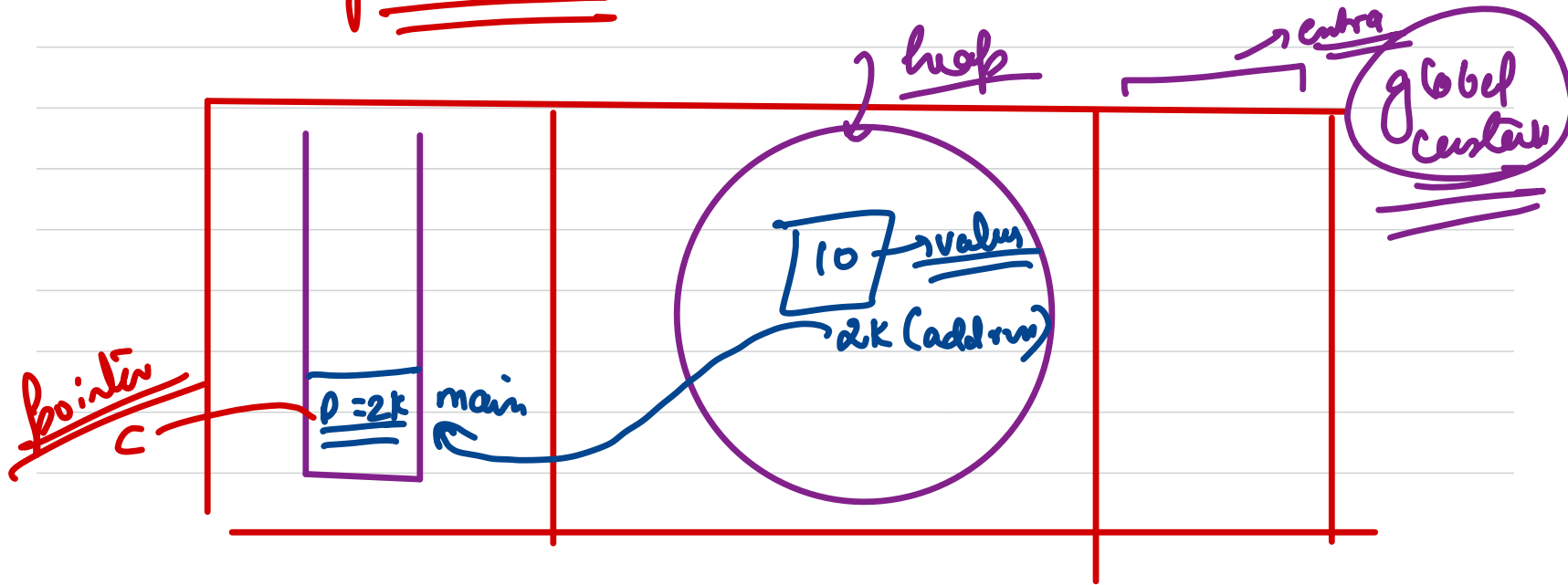
① function ends.

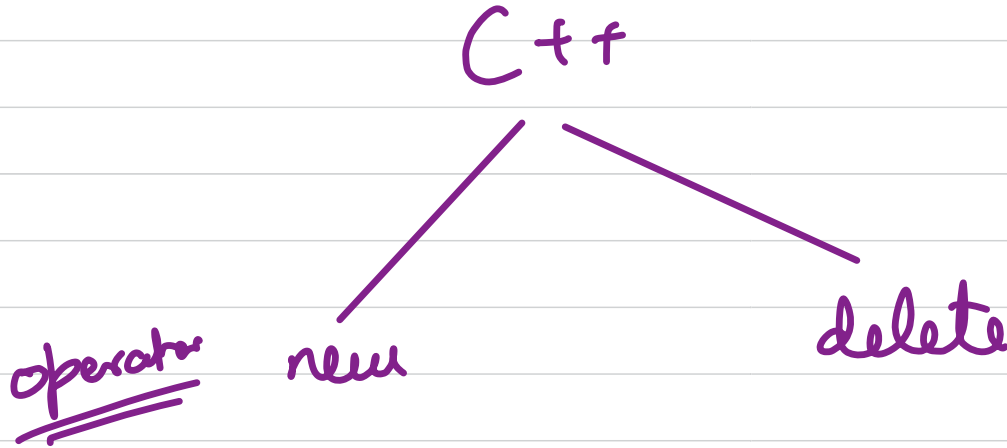
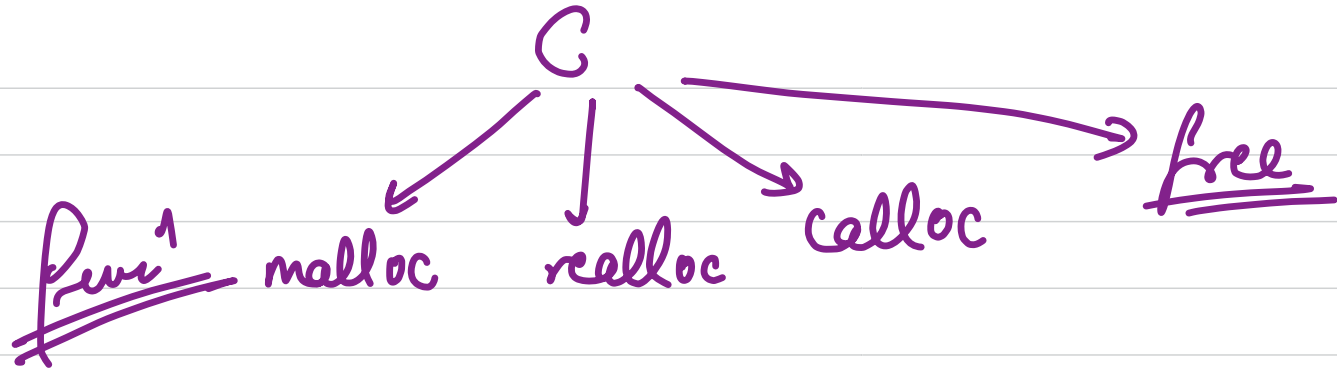
② function hits return

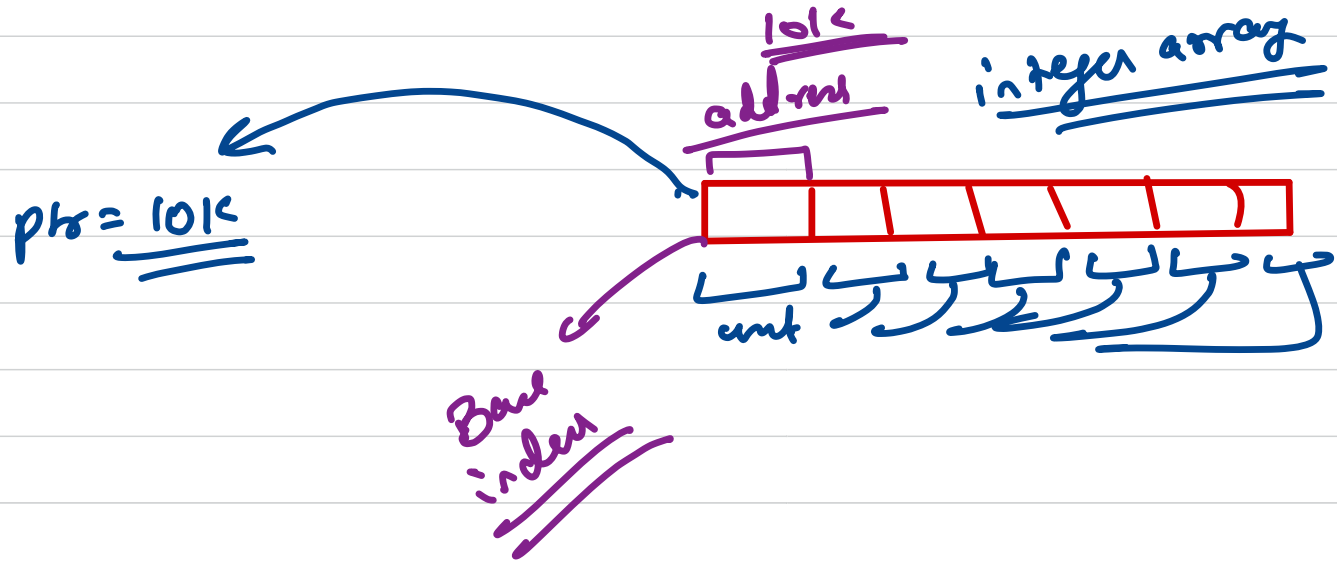
heap \rightarrow It is a big pool of memory.

\rightarrow In order to access heap, we need

pointers:







int * ptr = new int [7];

int arr[10];

Compiler resolves it

unslacked frames



Base index

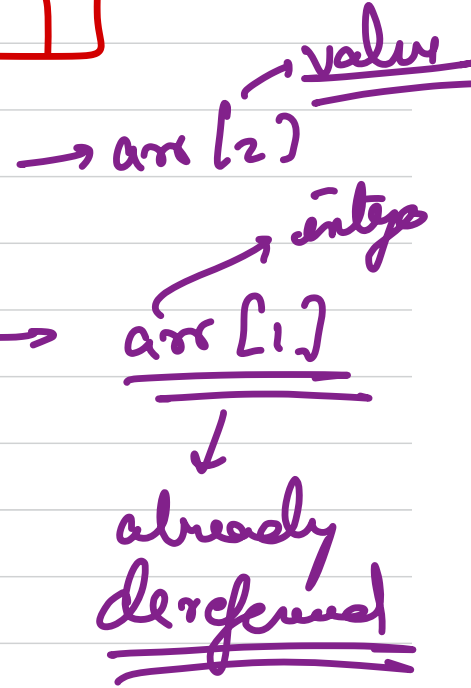
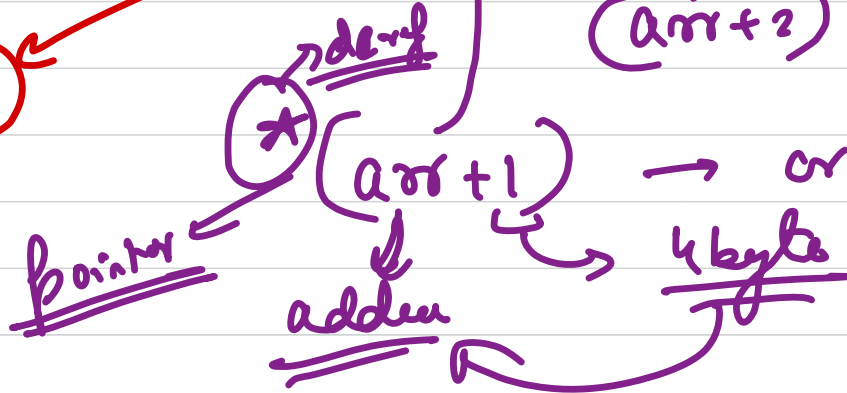
all block of int -

Name of the array (arr) is a pointer to base index.

0 based indexing



arr



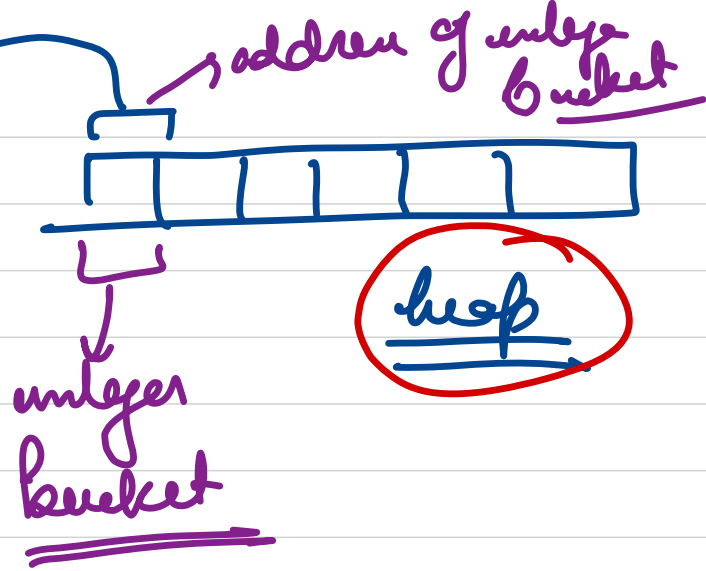
`int *arr = new int[5];`

this is resolved
at runtime.

`int *p = 10k`

address of
integer bucket

can be stored on
int pointer only.



$(p + x) \rightarrow$ pointer
 $p[x] \rightarrow$ value

few imp points

- ① arrays in C++ by default have garbage value (depends on compiler also)

```
int arr[100] = {0};
```

→ initialize every index with 0 value.

```
int *a = new int[10]();
```

`int a2[10] = {1};` → This will not init every index with 0.

unt $x[100] = \{1\};$

$x[0] \leftarrow \underline{\underline{1}}$

Q₂ Given an array of length n , with only 3 type of value $(0, 1, 2)$. These values are currently in random order. Sort the array in asc order. (You can't do counting, $TC \rightarrow O(n)$, No more than single pass allowed, no extra space). $(n \leq 10^8)$

$[1, 0, 1, 1, 2, 0, 1]$ $ans \rightarrow \underline{\underline{[0, 0, 1, 1, 1, 1, 2]}}$

exmp

[0, 0, 0, 0, 0, 1, 1, 1]_n

empty

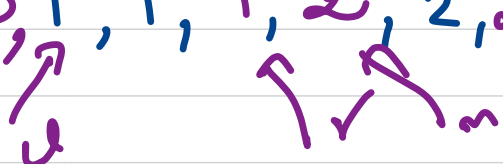
right left

all zeros all ones

2 after sorting

when we pass array in funcⁿ, we pass the address of the actual base address. So the function receives array by reference.

[0, 0, 0, 0, 1, 1, 1, 2, 2, 2]



DNF

Dutch
National
algo

flag

array
sort

all zeros

all ones

all twos