t1

hr
3

min
3

stack

count
2

amount
10

```
increaseHrs();

increasemins();
```

```
payingCar(){

}

nonPayingCar();
```

```
class ToolBooth{
int count
double amount;
int nonpayingcars;

}
```

Constant
- Variable as constant
- Pointer as constant

```
int n1 = 10;
const int *const ptr = &n1;
```

Constant
- Datamembers as constant
- Member Functions as constant
- Object as constant

```
const int n1 = 10;
const int *ptr = &n1;
```

```
const Test t1;
const Test *const this = &t1;
```

DataMember as constant

- If data member is made constant then it must be initialized inside the ctor members initializer list
- Once initialized we cannot change the value inside it.

Member Fuction as constant

- If member fuction is made constant then we cannot modify/change the sate of the current calling object
- Such functions are designed only to return the value or to display the state of object

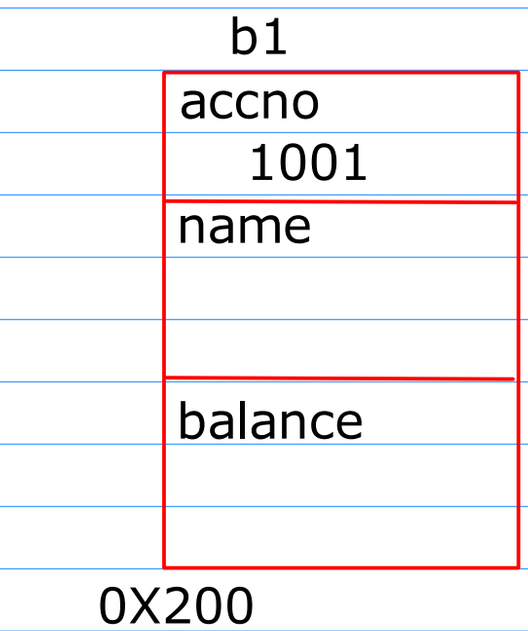
Object as constant

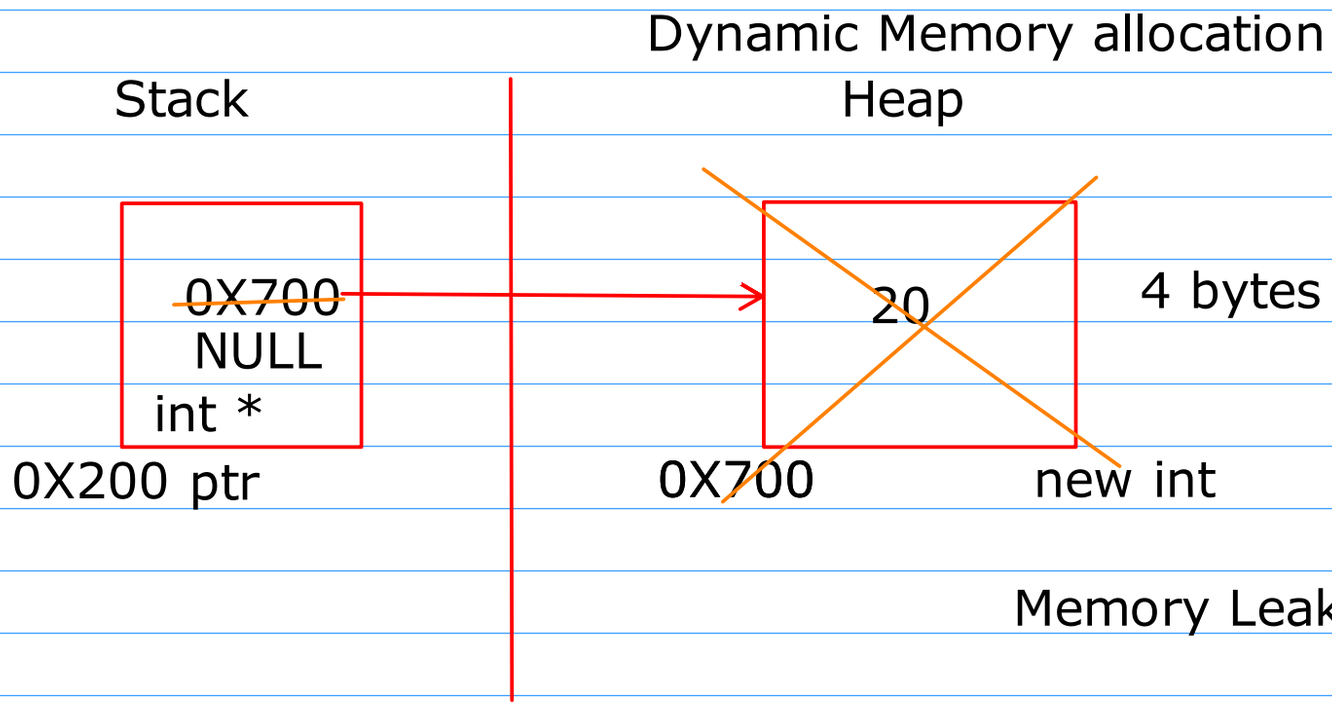
- Once the constant object is created we cannot modify its state.
- On constant objects we can call only constant member functions.
- We cannot call non constant member functions on constant object

```
const Test t1;
const Test *const this = &t1;
```

```
int n1 = 10;
const int *const ptr = &n1;
n1 = 100;
//*ptr = 200; // NOT OK
```

```
Test t1;
const * Test *const this = &t1;
t1.
```



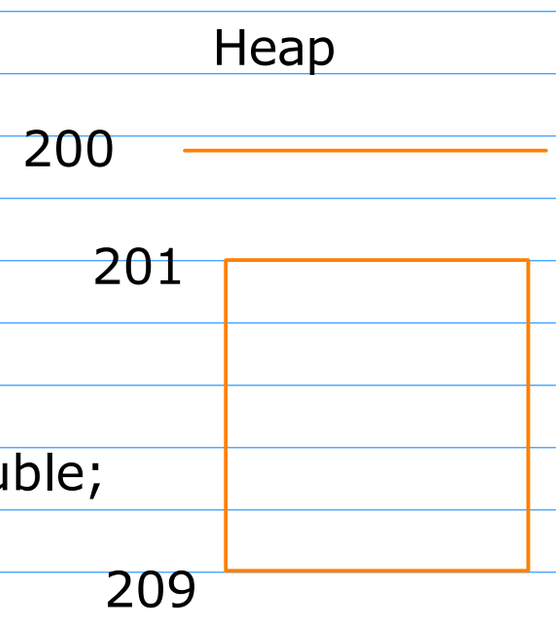

```
int *ptr = new int;
*ptr = 20;

*ptr -> 20
ptr -> 0X700 // Address on heap
&ptr -> 0X200 // Address of ptr

delete ptr;
delete 0X700;
ptr = NULL
```

```
double *ptr = new double()
Time *tptr = new Time();

delete ptr;
delete tptr;
```

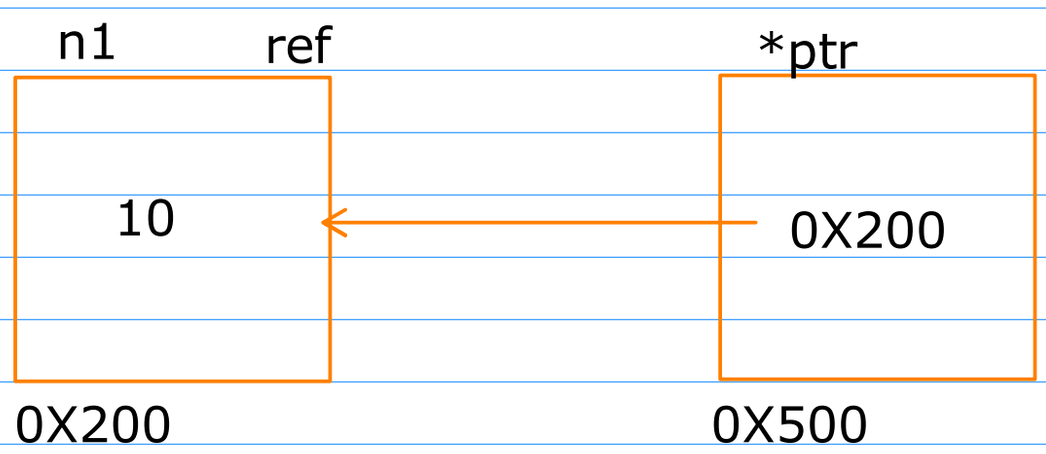



```
new int[5]

//ptr = NULL

int *ptr = 200
//delete ptr
double *ptr2 = new double;
//ptr2 = 200
if(ptr!=NULL)
        cout<<*ptr;
```

Dynamic memroy using malloc()
free()




&ref = 0X200
ref = 10

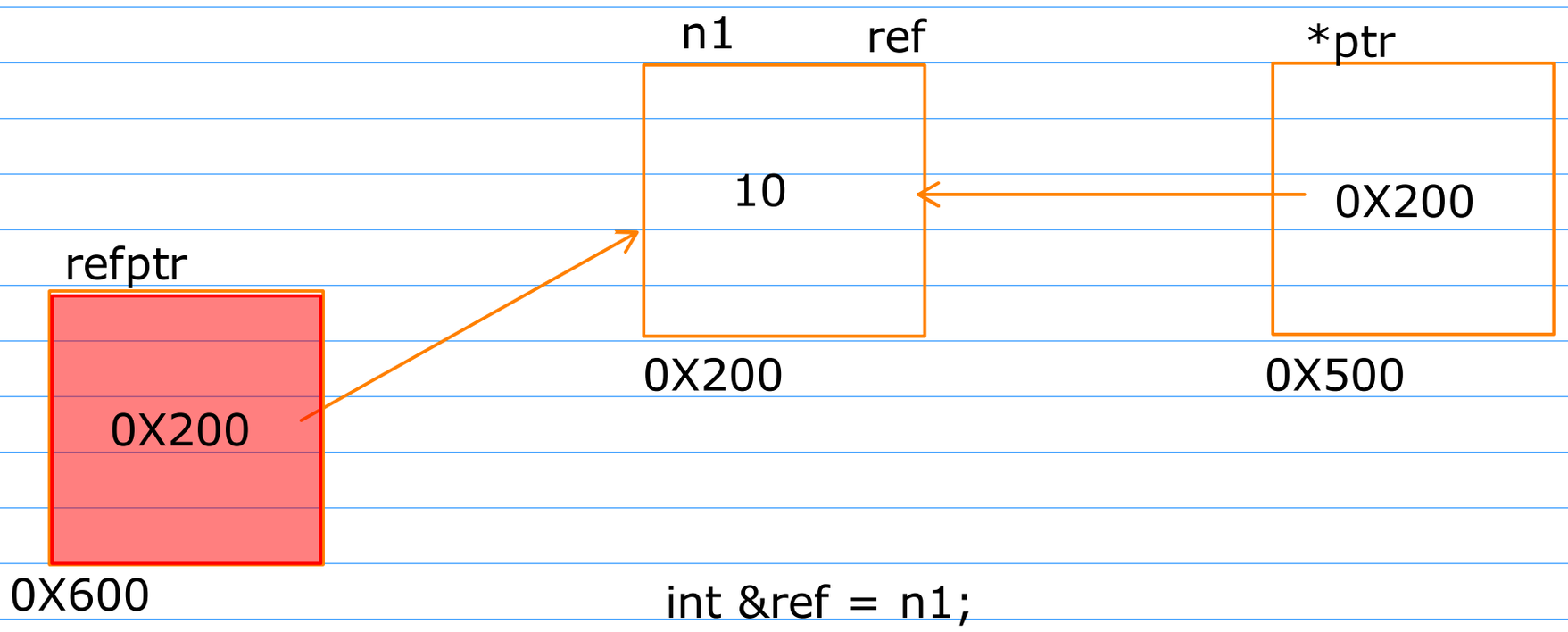
&n1 = 0X200
n1 = 10

int *ptr = &n1

ptr -> 0X200
&ptr-> 0X500
*ptr -> 10

int &ref = n1;

Reference
- It is an alias for an existing memory location
- It is internally a constant pointer
- It is used to simplify the pointer syntax




int &ref = n1;

Internally it is
a constant pointer

ref = *refptr;
&ref = refptr;

static gloabal variables
static local varaiables

Static
1. Data members
2. Member functions

Static Data Members
- static means sharing. If we want to share the value of data member across multiple objects then we need to declare them as static
- static data members will get space on the data section
- These needs to be initialized outside the class on global scope using acope resolution operator (::)

t1
n1
2
n2
3
8 bytes

t2
n1
5
n2
6
8 bytes

t3
n1
n2
8 bytes

Stack

4 7
n

Data

c1
radius
5
PI
3.14
16 bytes

c2
radius
7
PI
3.14
16 bytes

c3
radius
9
PI
3.14
16 bytes

c1
radius
5
8 bytes

c2
radius
7
8 bytes

c3
radius
9
8 bytes

Stack

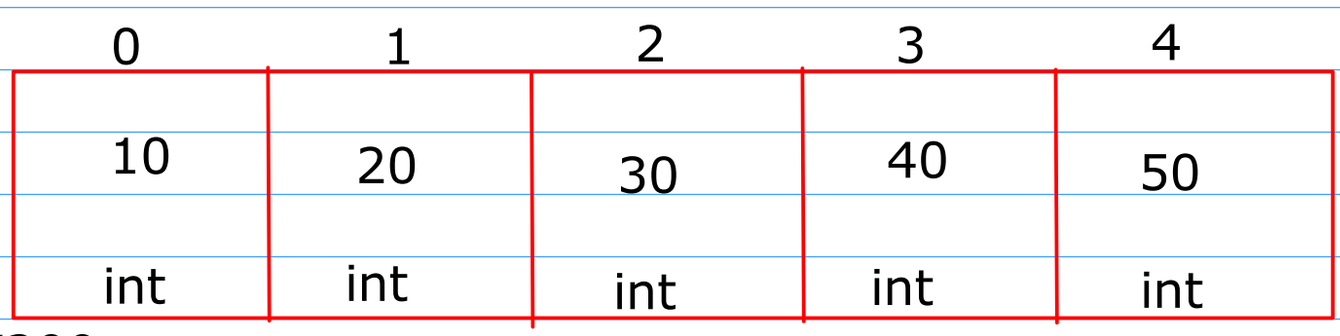
PI
3.14
8 bytes

Data

Array

- It is a Data structure used to store the elements of similar type in contigious memory location
- The size of array is fixed
- The elements can be accessed using index
- Types of Array
    1. Single Dimensional Array
    2. MultiDimensional Array

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




int arr[5];

0X200

arr

0X200[0] = 10
0X200[1] = 20
arr[2] = 30
arr[3] = 40
arr[4] = 50

Employee arr[5]