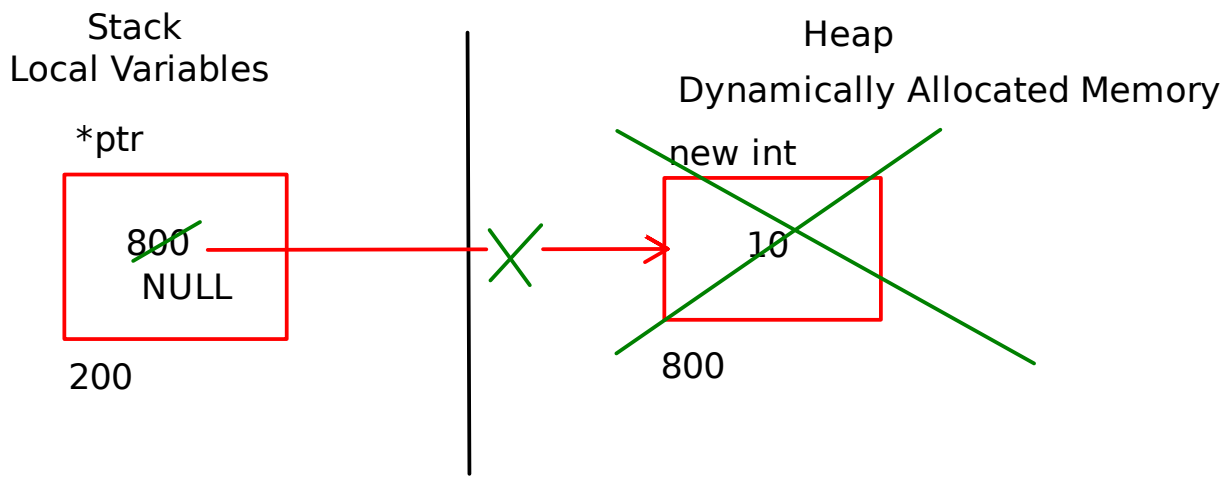
```
class Point{
int x;
int y ;
Point(intx,int y):{
//parameterized
this->x = x;
this->y = y;
}

Point(){

}
};


main(){

Point p1(1,2);

Point p2;

}
```

```
const int num1 = 10;

const int *const ptr = &num1;
```

```
class Test{
int num1;
const int num2;

Test():num2(10)

}

void dispay() const
{
//num1 = 100//not ok
}

main(){
const  Time t;
}
```



?-> how much memory
?-> for which type

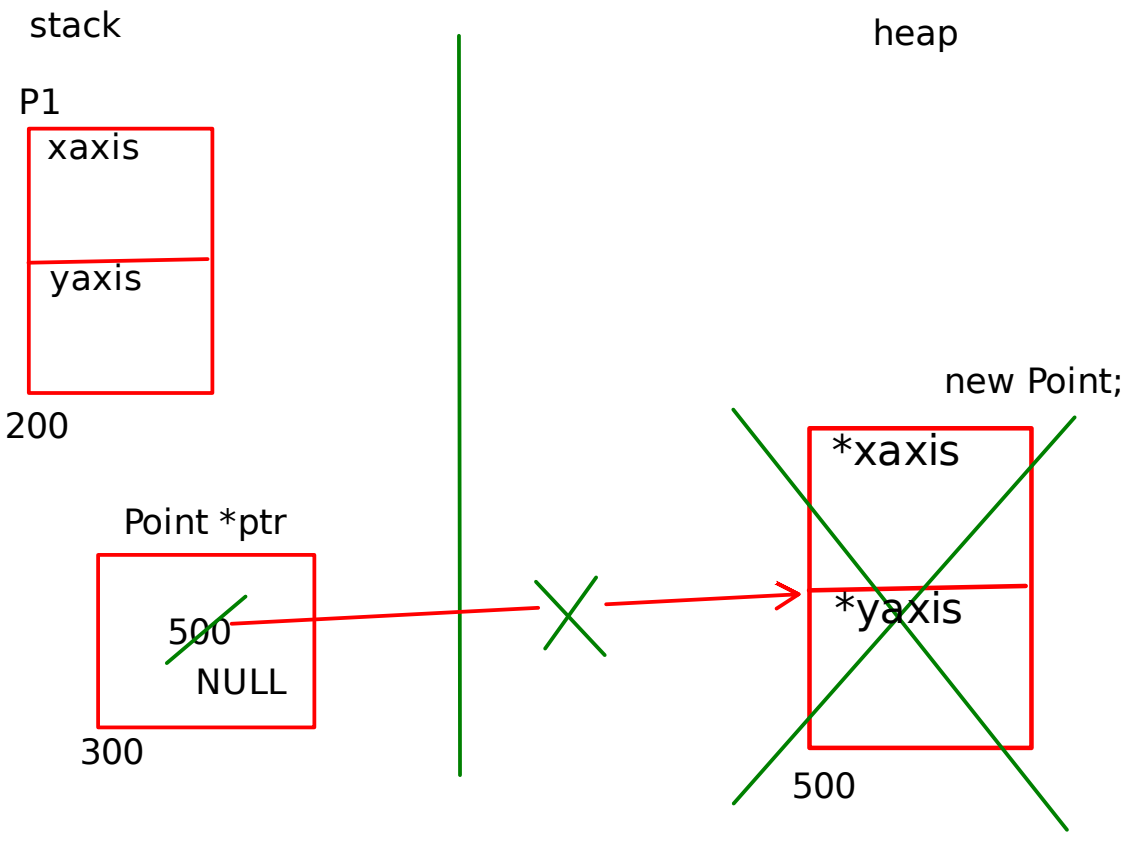
```
  int main(){

  int *ptr;
  ptr = new int;
  *ptr =10;
  delete ptr;
  ptr = NULL;

  return 0;
  }
```

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

valgrind -> used to detect memory leakages in the program

sudo apt install valgrind;

```
int *ptr = new int(10);
cout << "Value at *ptr = " << *ptr << endl;
delete ptr;
```



delete ptr;

```
void swap(int n1, int n2){
        int temp = n1;
        n1=n2;
        n2 = temp;
}


int main(){
int n1=10;
int n2 = 20;
swap(n1,n2); // pass by value
cout<<"n1 = "<<n1; //10
cout<<"n2 = "<<n2; //20
return 0;
}
```

```
void swapByAddress(int *ptr1,int *ptr2){
int temp = *ptr1;
*ptr1 = *ptr2;
*ptr2 = temp;
}


int main(){
int n1=10;
int n2 = 20;
swapByAddress(&n1,&n2); // pass by address
cout<<"n1 = "<<n1; //20
cout<<"n2 = "<<n2; //10
return 0;
}
```

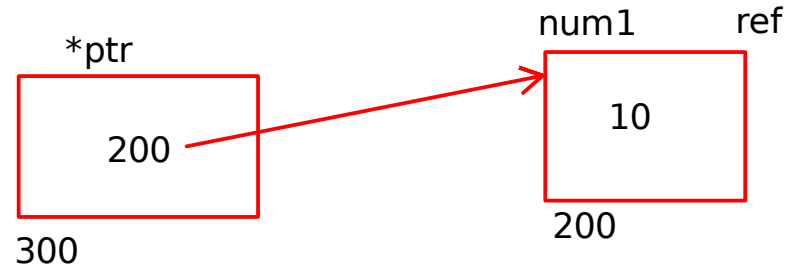
references

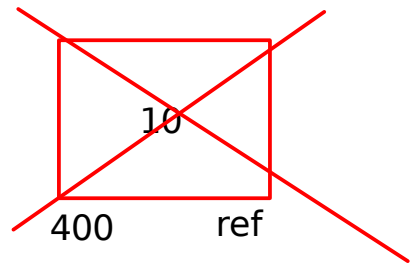
```
void swapByReference(int &ref1, int &ref2)
{
int temp = ref1;
ref1 = ref2;
ref2 = temp;
}


int main(){
int n1 = 10;
int n2 = 20;

// pass by refrerence
swapByReference(n1,n2);

cout<<"n1 = "<<n1; // 20
cout<<"n2 = "<<n2; //10
}
```

Reference
It is an alias given to an exixting memory location

```
int main(){
int num1 = 10;
int *ptr = &num1;

cout<<ptr;  //200
cout<<&ptr; //300
cout<<*ptr; // 10

cout<<num1; //10
cout<<&num1;//200
}
```

```
int main(){
int num1 = 10;

int &ref=num1; // reference

cout<<ref; // 10;
cout<<&ref; // 200;

cout<<num1; // 10;
cout<<&num1; // 200;
}
```





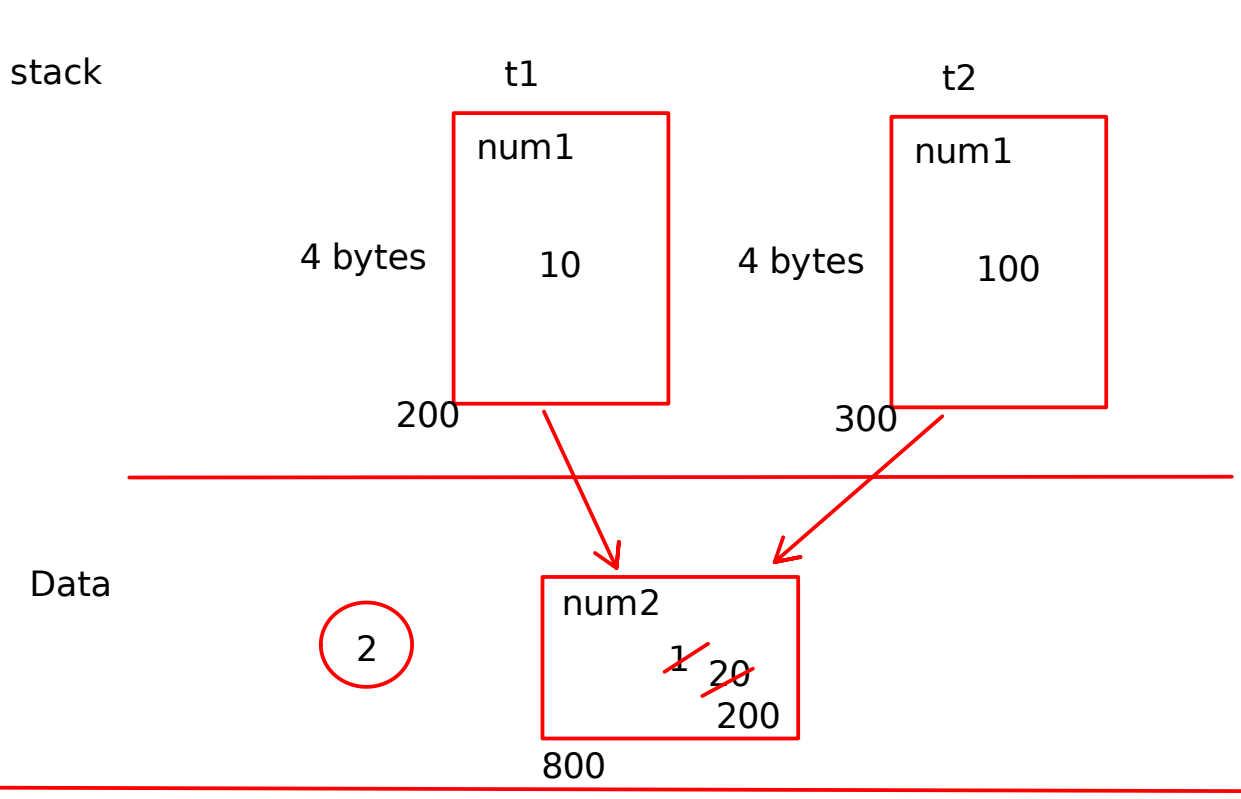
Static

static variables are also called as shared variables

static data members
static member functions

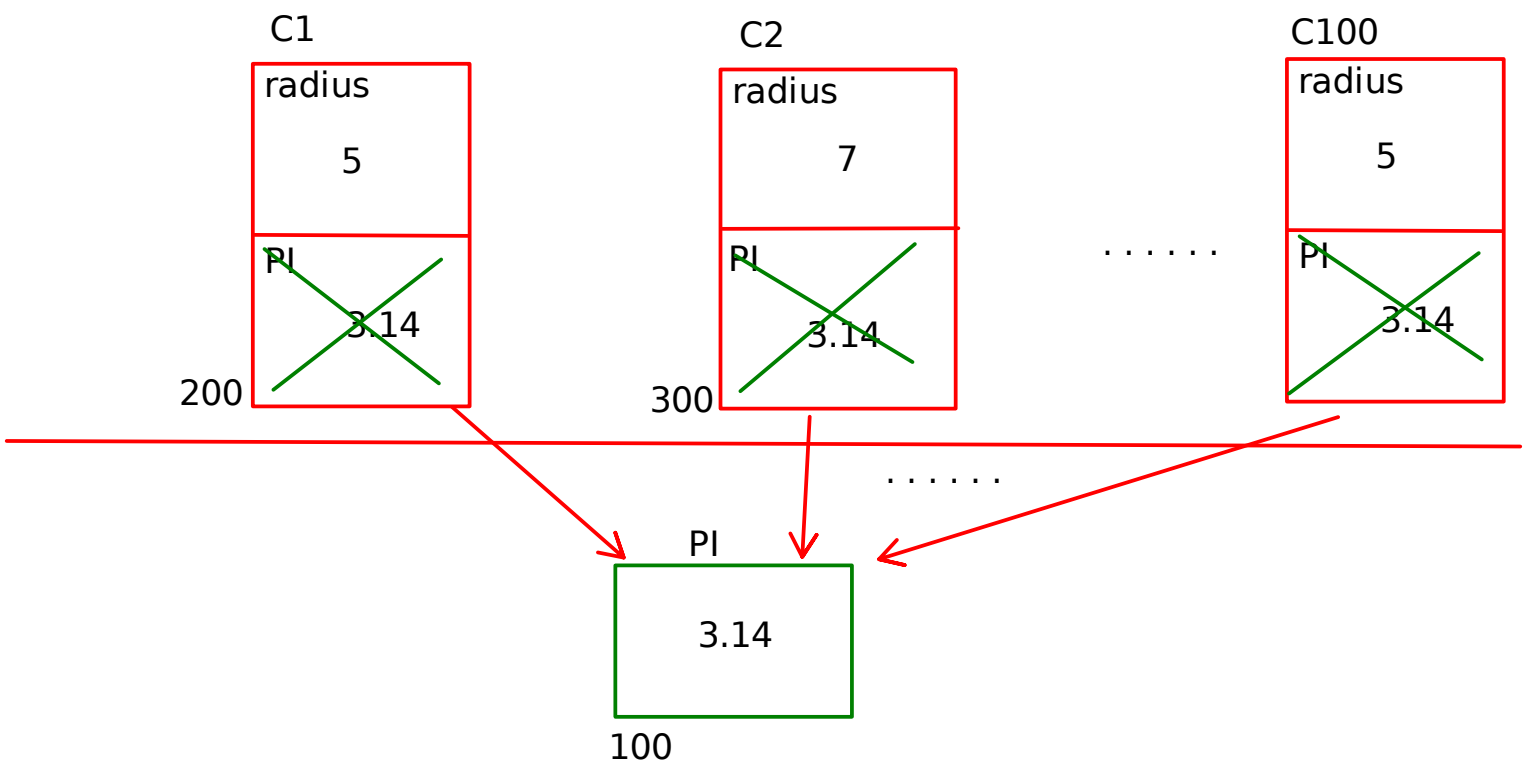
```
main(){
Test::displayNum2();
}
```

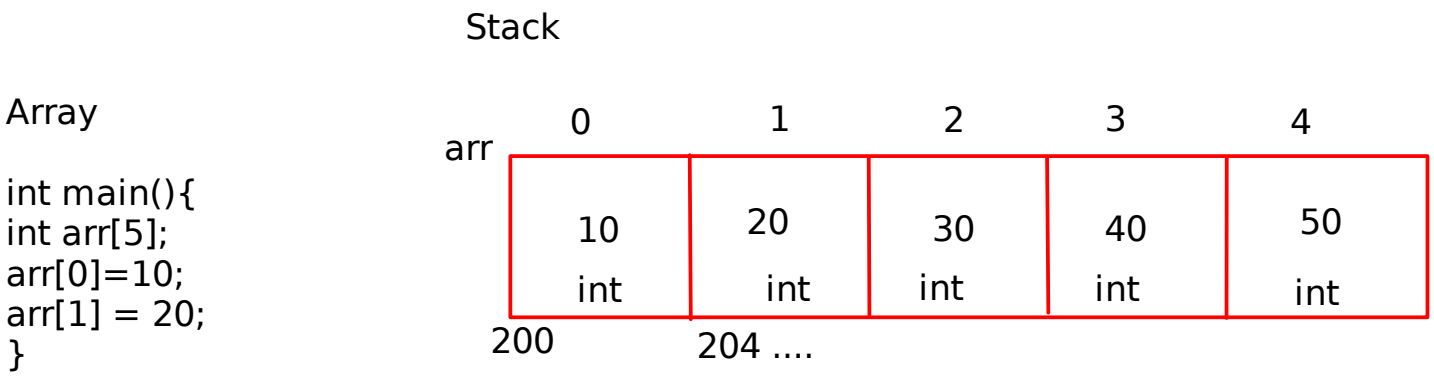
test t1;
1. Memory allocation on stack ->(num1,num2)
2. ctor call (num1 =10, num2 =20)

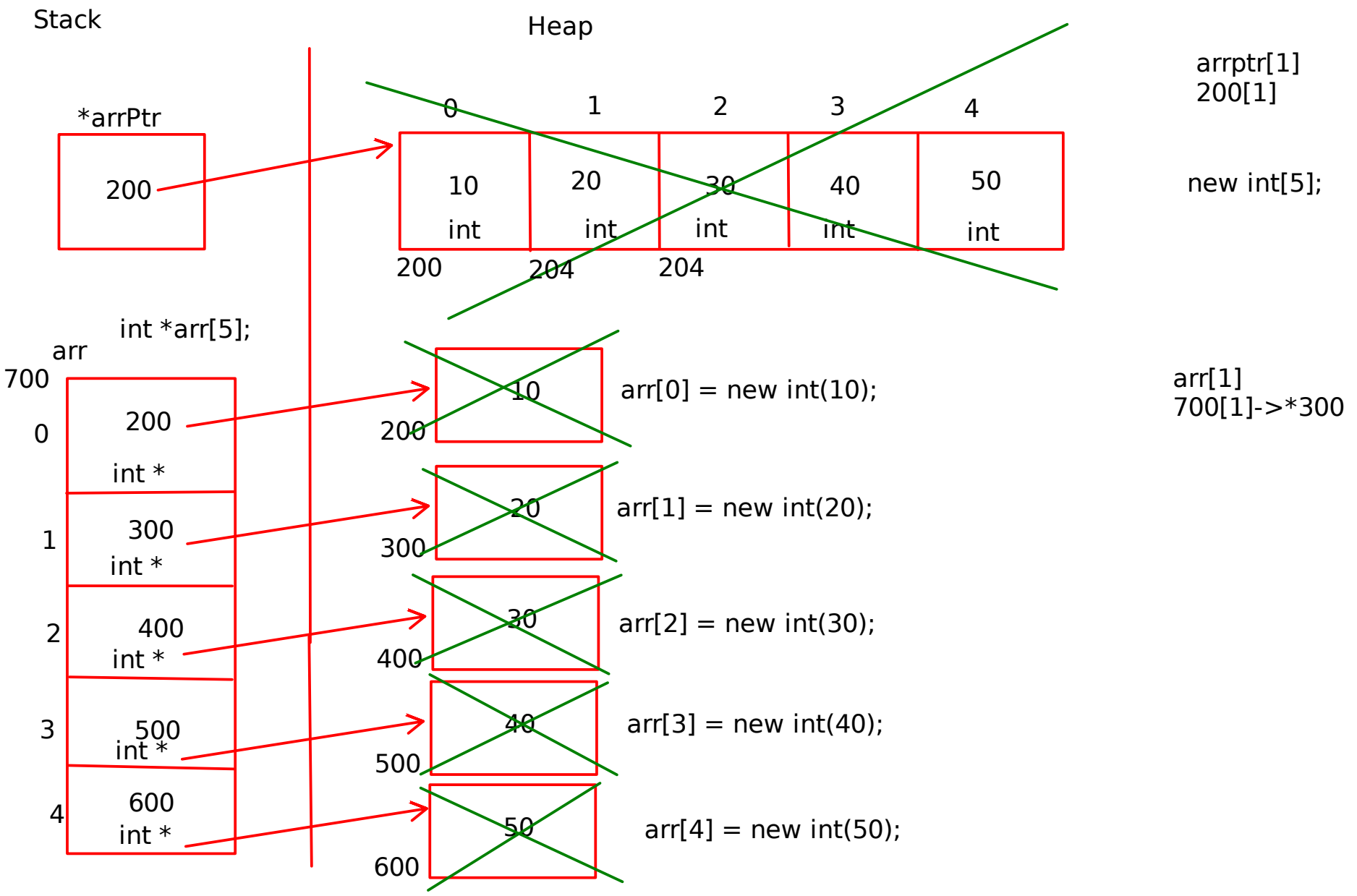


text 1

```
void displayTest() // Test *const this
  {
     cout << "num1 = " << num1 << endl;
     cout << "num2 = " << num2 << endl;
  }

 static void displayNum2()// Nothing
 {

 }
```


C1
radius
5
PI
3.14
200
C2
radius
7
PI
3.14
300
C100
radius
5
PI
3.14
. . . . . .
. . . . . .
PI
3.14
100


Stack
Array
0
1
2
3
4
arr
int main(){
int arr[5];
arr[0]=10;
arr[1] = 20;
}
10
int
20
int
30
int
40
int
50
int
200
204 ....


Stack
Heap
*arrPtr
200
0
1
2
3
4
10
int
20
int
30
int
40
int
50
int
200
204
204
arrptr[1]
200[1]
new int[5];
int *arr[5];
arr
700
0
200
int *
1
300
int *
2
400
int *
3
500
int *
4
600
int *
10
200
arr[0] = new int(10);
20
300
arr[1] = new int(20);
30
400
arr[2] = new int(30);
40
500
arr[3] = new int(40);
50
600
arr[4] = new int(50);
arr[1]
700[1]->*300