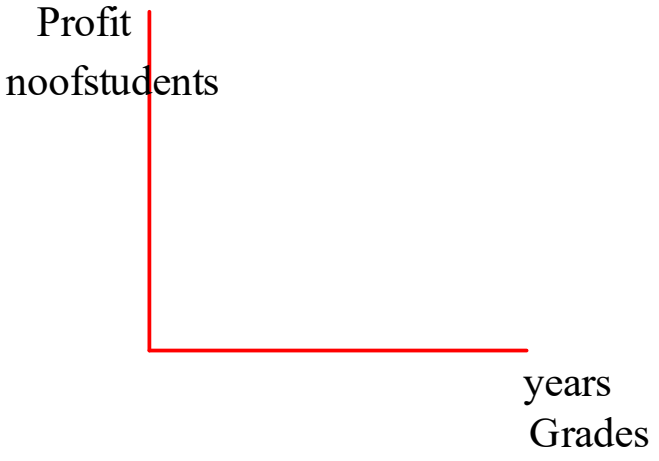
Template
- Function Template
- Class Template

```
template<typename X>
void swap(X *ptr 1, X *ptr2){

}

double n1;
double n2;
swap(&n1,&n2); // type inference
swap<double>(&n1,&n2);
```

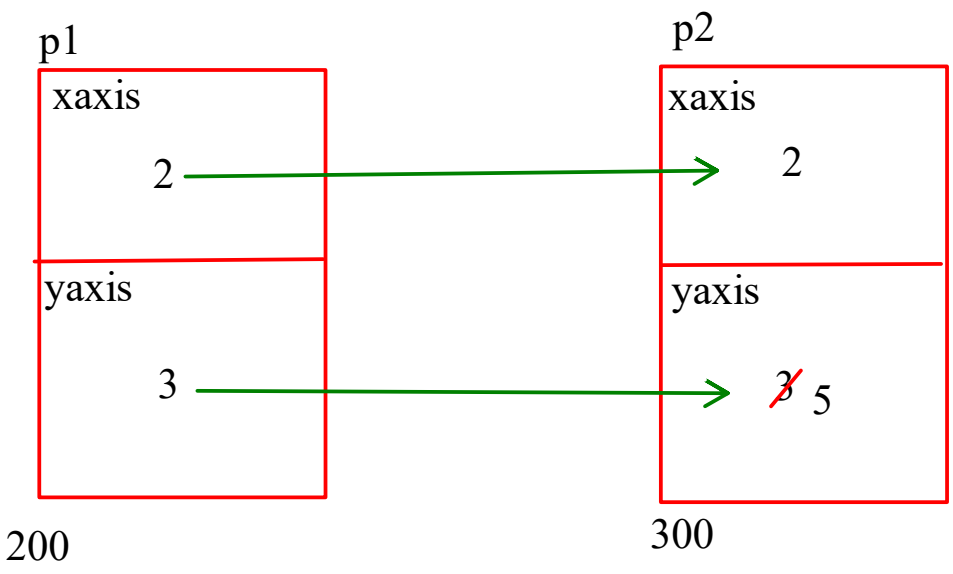


Stack

```
Employee arr[5];
Employee e1(1,"anil",10000);
arr[0] = e1;

Employee *arr[5];
arr[0] = new Employee(1,"anil",10000);
```

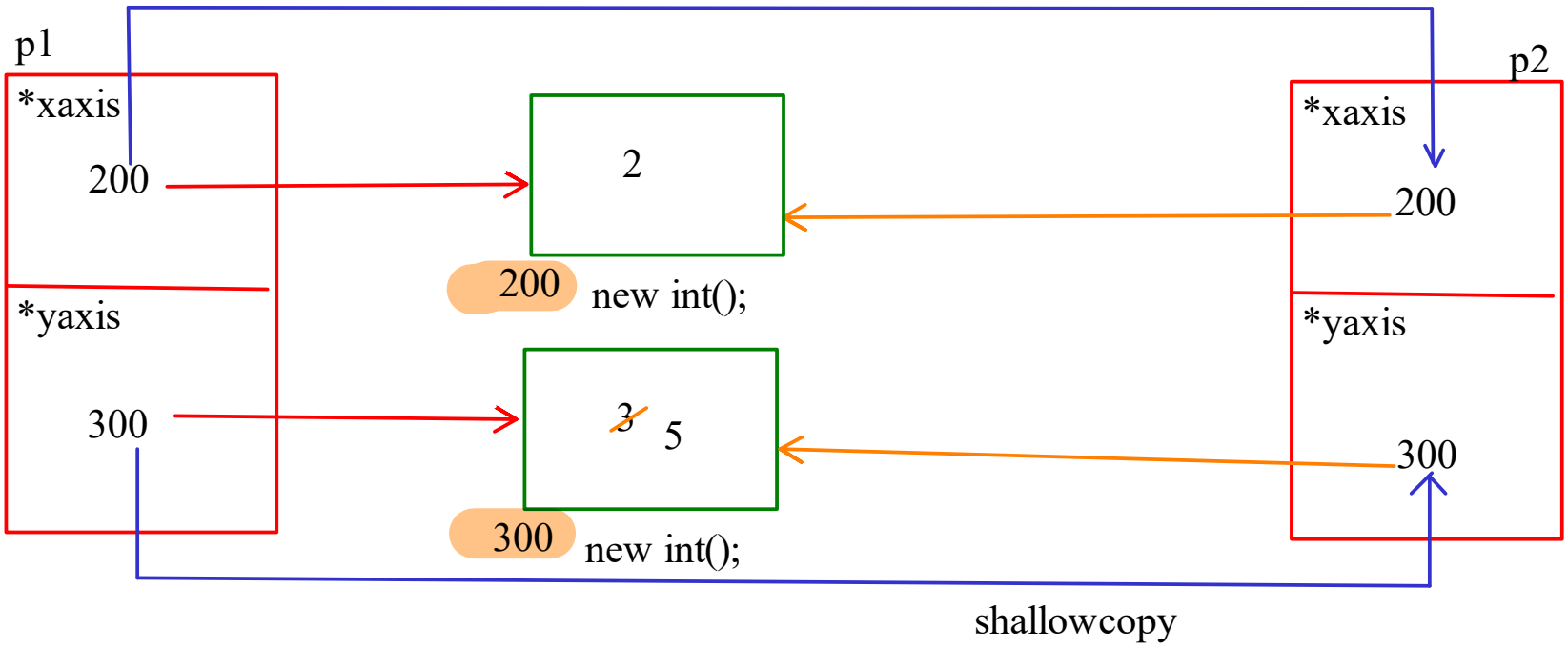
```
Employee e1(1,"Anil",10000); // parameterized
Employee e2; // Parameterless Ctor
Employee e3 = e1; // copy Ctor
```



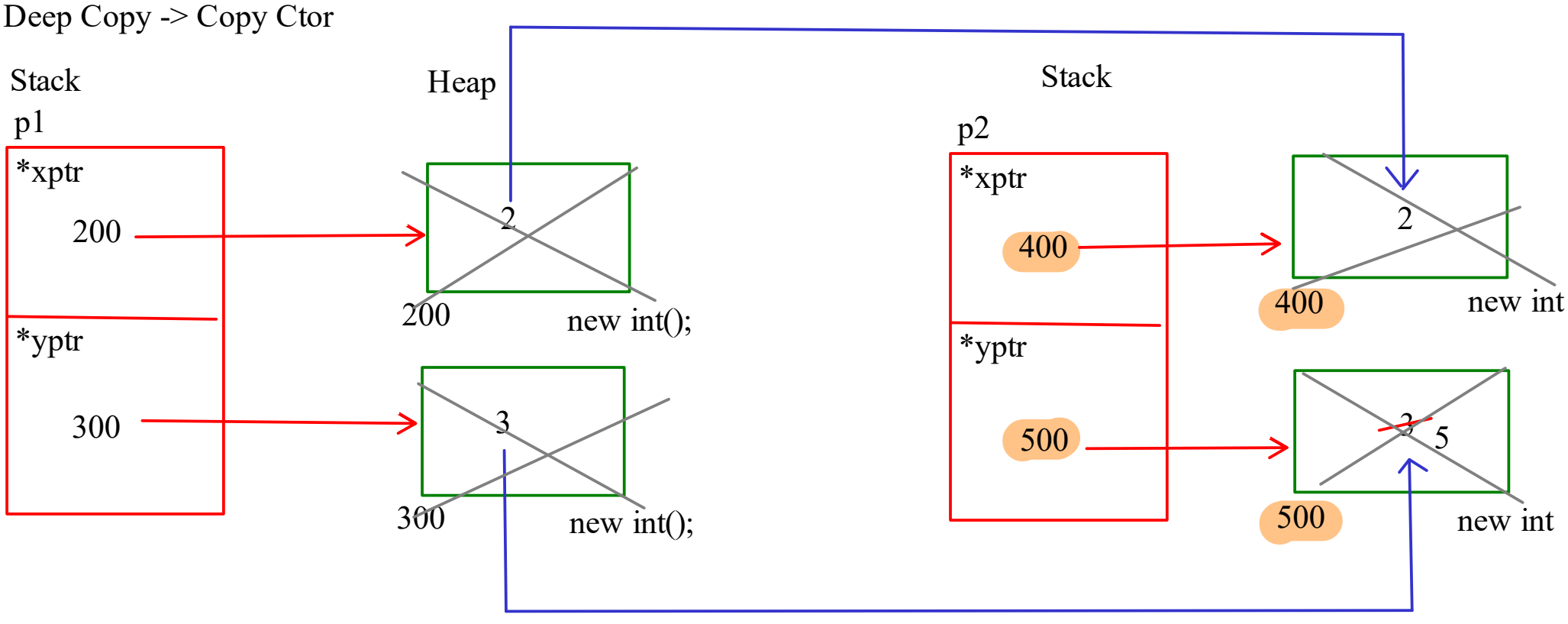
Stack

Heap

Stack



Deep Copy -> Copy Ctor



Template
1. Function
2. class

```
template<typename T>
template<class T>
template<class X, class Y>
```

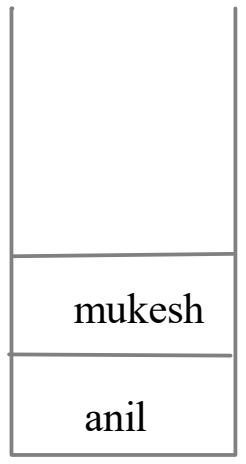
```
fl<int>(10,20);

Point<int,int> p1(1,2);
```

```
template<class T >
class Stack{

T *ptr= new T[5];

}
```




```
Stack<int> s1;
// int* ptr = new int[5];

Stack<char> s2;
//char* ptr = new char[5];

Stack<Employee*> s3;
// Employee** ptr = new Employee*[5] ;
```

```
Point p1; // Parameterless Ctor;
Point p2(2,3); // Parameterized Ctor
Point p3 = p2; // Default copy ctor -> Shallow copy
```

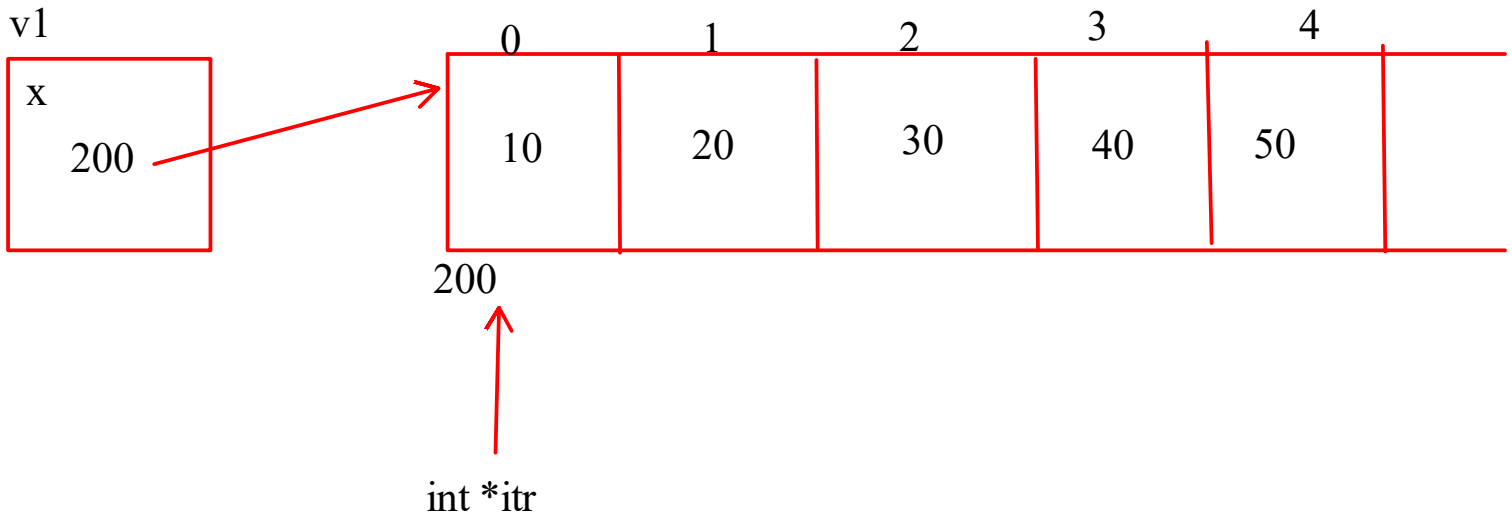
Copy Ctor
- It is a ctor that gets called when we try to initialize the object with an already created object.
- If your class does not provide the copy ctor then compiler adds one copy ctor called as default copy ctor.
- Default copy ctor does the shallow copy.
- in shallow copy the state of an object is copied as it is in the new object. i.e the value(values or address) are copied as it is.
- If the class consists of pinter type of data members and dynamic memory allocation is done then both the object states will point at the same memory on the heap section
- To avoid this we need to perform deep copy.
- To perform deep copy we can define our own copy ctor.
- Copy ctor is a single parameterized ctor with the parameter of the same type as of the class, which accepts the object by refrence.

STL- Standard Template Library
- It has 4 components

1. Containers -> Data Structures for the data Storage
2. Algorithms
3. Function Objects (Functors)
4. Iterators

Sequence Containers
- vector

Sequence Adapter Containers
- queue
- stack


v1
x
200
0
1
2
3
4
10
20
30
40
50
200
int *itr