Pointers as data members:

1) Constructor (Data / param / Allocation)
2) Destructor (deallocation)
3) Copy constructor (deep copy)
4) Assignment (deep copy)

```
class Student {
    int rollnum;
    int size;
    int *marks;
public:
};
```

*marks → Data member

```
Student S1;
int arr[3] = {3, 5, 100};
Student S2(1234, arr);
Student S3(1234, arr, 3);
```

```
Student(int rn, int arr[3]);
Student(int rn, int *arr, int s)
{
    rollnum = rn;
    size = s;
    marks = new int[size];
    loop arr to marks;
}
```

Arrays of objects.

Static

```
Student sarr[10];
```

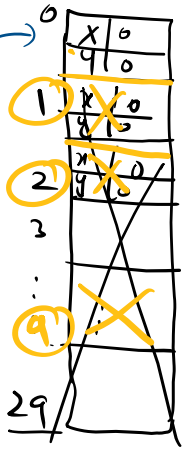
Simple loop

```
for(int i=0; i<10; i++)
{
    sarr.printStudent();
    sarr[i].printStudent();
}
```

```
Point arr[10] = {Point(2,3), Point(5,1), Point(6,8), Point};
```

member initializer

Dynamic Arrays of objects




```
Point *ptr = new Point[30];
```

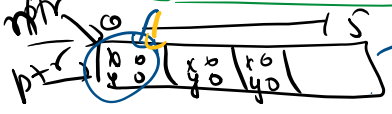
① Called default constructor

```
for(i=0->29)
{
    ptr[i].printPoint();
}
```

② ① Extend ② Shrink } to is

```
delete[] ptr;
```

```
class Point
{
    int x, y;
public
    ...
};
```

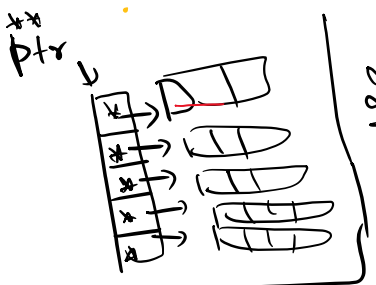


→ Slow
→ copy of too many objects. (8 bytes)

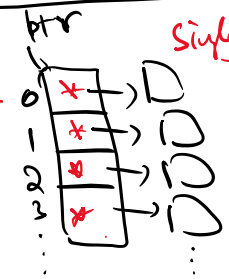
→ inefficient

③ remove objects.
① ② (9) 30
not possible

Account, Student (64 bytes)



float



Single object

Point ** dp = new Point*[30];

for(int i=0; i<10; i++)
{
fin >> a >> b;
dp[i] = new Point(a,b);
}

file

parameterized

for(int i=10; i<19; i++) copy constructor *

dp[i] = new Point(*(dp[i-10]));

i = 10

(*(dp[0]))

Point(const Point & p);

for(int i=20; i<29; i++)

dp[i] = new Point; default constructor

Extend Shrink

4 bytes

dp is dp

shrink.

copy Address.

is new

extend 30

Remove any object from array.

delete dp[1];
dp[1] = nullptr;
delete dp[2];
dp[2] = nullptr.
delete dp[9];
dp[9] = nullptr;

for(int i=0; i<30; i++)
{ if(dp[i] != nullptr)
dp[i]->printPoint();
dp[i]->print();
*(dp[i]).print();

**dp

data intPoint, Student

2d

***tp

2d

4x4 16

8*8 64

Address.

helpful