CS 32 – Discussion 1B – Week 1

TA: Hadley Black , LA: Jim Zhou

OH: M 2:30–4:30pm
T 4:30 – 5:30 pm
F 3:30 – 4:30 pm

Discussion:

1. 20 – 40 minutes of recap lecture.
2. Work on LA worksheet in breakout rooms.

Pointer Review:

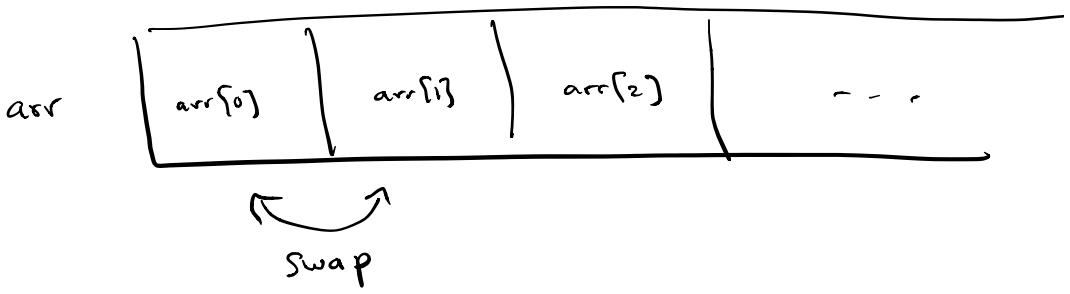
Resource: class website → Carey Nachenberg's Slides
→ lecture 3

Pointer: a "variable" which holds the address of another variable

Big picture: Give a "lightweight" / "efficient" way of accessing / manipulating data.

Scenario : Many instantiations of a class C.

Approach 1 : Put them all in an array

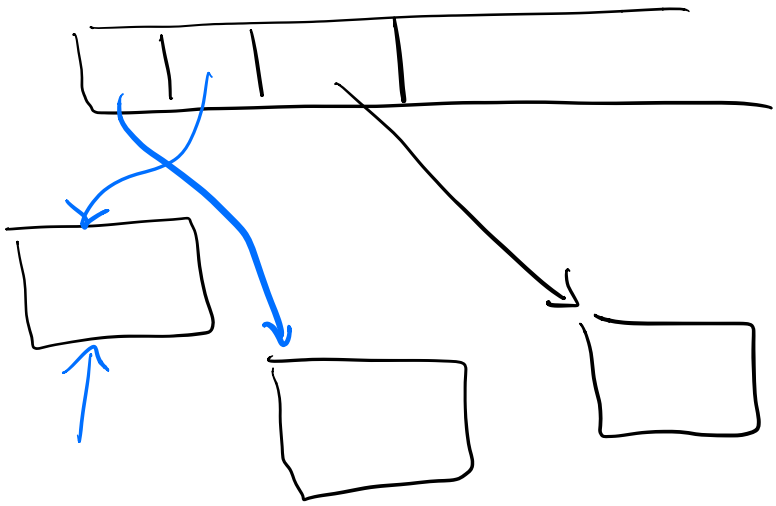



```
C temp = arr[0]
arr[1] = arr[2]
arr[2] = temp
```

copy a C object 3 times

expensive

Approach 2 : array of pointers



```
C* temp = arr[0]
arr[1] = arr[2]
arr[2] = temp
```

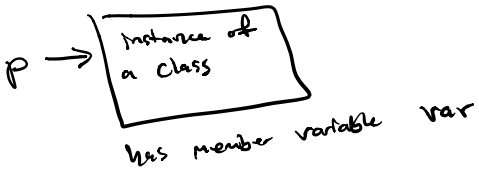
Basic syntax:

Declaration: `int* p = nullptr;`

Dereferencing: `*p` (returns what is pointed to by p)

Equivalent → `p->var`

Equivalent → `(*p).var`



Address-of: `int var = 10;`

`int* p = &var;`

↑ returns address of var

`cout << *p;` ← return 10

`*p = 20;` (equiv. `var = 20;`)

`cout << *p;` ← return 20;

New & delete:

new keyword: allocates memory, calls constructor, returns a ptr to the instantiated object.

`int* p = new int(10);`

`int(10)`: Constructor for int type

`*p = 20;`

Delete keyword : calls destructor, deallocates memory

```
int var = 10 ;
int* p = new int(10) ;
```

Pass-by-reference :

```
void foo (int &val)
{
    val = 5 ;
}

int main()
{
   int x = 1 ;
   foo(x) ; <-
   cout << x ; <- return 5
}
```

```
void foo ( int *ptr)
{
   *ptr = 5 ;
}

int main ()
{
   int x = 1 ;
   foo(&x) ;
   cout << x ; <- return 5
}
```