C++ V

Memory and Pointers

Memory Addresses

- Variables are stored in memory
- The memory used has an address
  - Much like a safe deposit box
  - The address is stored in another box
- Declaring a normal variable instructs the compiler to reserve some memory (wherever it decides)
- Using the variable gets what what put in the above memory
- What if you just want to get the memory location and do something with it?

Getting an Address: &

- `int x = 12;`
- We can get the address of x with &x. We can print it out to see it (see demo).
- To store it in another variable, we need the other variable to be a *pointer*
- `int *y = &x;`
- Then `y` will have the address. To get the value in the address, use `*y`

Pointers example

On the right:

1) `&x` gives you the memory location of the variable `x`

2) `*p` gives you a pointer. `p` then gives you a memory location (we must put some memory into p first; see demo)

`*` dereferences a pointer

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(){
6      int x = 5;
7      int *p;
8
9      p = &x;
10     cout << "mem location of x: " << &x << endl;
11     cout << "value of p: " << p << endl;
12     cout << "value of *p: " << *p << endl;
13
14     return 0;
15 }
```

```
mem location of x: 0x7fff50cfcb78
value of p: 0x7fff50cfcb78
value of *p: 5
```

Exercise:

What would be the result of this program? Discuss in groups of 2 or 3

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(){
6      int a = 5, b = 10;
7      int *p1, *p2;
8      p1 = &a;
9      p2 = &b;
10     *p1 = 10;
11     p1 = p2;
12     *p1 = 20;
13     printf("a = %d\n", a);
14     printf("b = %d\n", b);
15
16     return 0;
17 }
```

Pointers to Pointers

`paul` points to melissa's memory, `ramon` points to paul's memory

`paul` is a variable storing a memory location, it has a memory location too

```
1  #include <fileiostream>
2  using namespace std;
3
4  int main(){
5
6      int **ramon;
7      int *paul;
8      int melissa = 5;
9
10     paul = &melissa;
11     ramon = &paul;
12
13     printf("ramon = %d\n", ramon);
14     printf("&paul = %d\n", &paul);
15     printf("*ramon = %d\n", *ramon);
16     printf("&melissa = %d\n", &melissa);
17     printf("**ramon = %d\n", **ramon);
18
19     return 0;
20 }
```

Arrays as Pointers

- `*array` is a pointer to a contiguous set of memory locations
- array indices get the contents of the array at each of these locations
- this pointer can be reassigned to different memory, but a normal array cannot

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5
6      int *array = new int[10];
7      for(int i = 0; i < 10; i++){
8          array[i] = i + 100;
9          printf("val = %d\n", array[i]);
10     }
11
12     cout << array << endl;
13
14     return 0;
15 }
```