CS 225 Spring 2019 :: TA Lecture Notes
1/18 Memory

By Wenjie

Thanks to all of you who pointed out typos in the notes, and gave formatting suggestions:)
But the google doc seems auto-reject comments from time to time - please lmk if you ran into this!

- **Variable / reference variables / pointer**

```
1  Cube    s1  // a variable containing a Cube object
2  Cube &  s2; // a reference to a variable of type Cube
3  Cube *  s3; // a pointer to a variable of type Cube
```

- **Pointers**
  - Stores a memory address of the instance instead of storing data
  - Must resolve the memory address to access the data
  - Pointers are extremely powerful and extremely dangerous

main.cpp

```
1  int main() {
2     cs225::Cube c;
3     std::cout << "Address storing `c`:" << &c << std::endl;
4
5     cs225::Cube *ptr = &c;
6     std::cout << "Addr. storing ptr: "<< &ptr << std::endl;
7     std::cout << "Contents of ptr: "<< ptr << std::endl;
8  }
```

- **Indirection operators**
  - **&c** - returns the memory address of c's data. We can say that & operator takes us one step away from the data.
  - ***ptr** - returns the data at the memory address contained at ptr, aka dereferencing a pointer. We say that * operator takes us one step closer to the data.
  - **ptr->**
    - (*ptr).getVolume() = ptr->getVolume()

CS 225 Spring 2019 :: TA Lecture Notes
1/18 Memory

By Wenjie

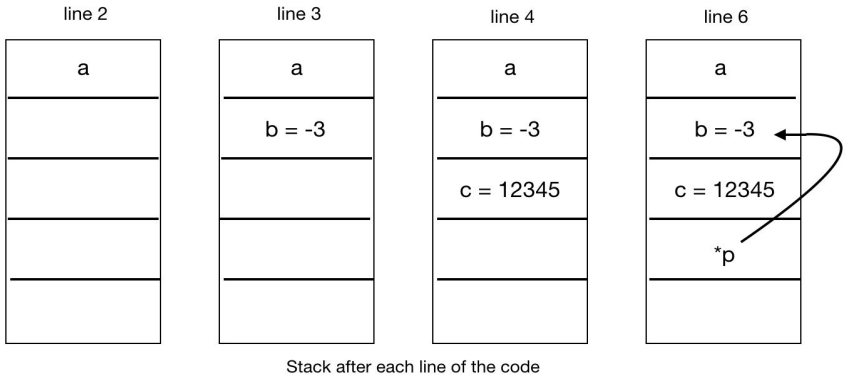
- **Stack Memory**
  - The default type memory
  - Starts near the top of memory (but it does not necessarily start with the most top piece memory)
  - Starts at a high address and it grows towards 0
  - The data is read from low to high (the data is read up).
  - All variables are by default on stack (automatic variables)
  - Function `sizeof()` : return the size of a type in bytes
    - Int variable takes 4 bytes
    - Pointer takes 8 bytes

example1.cpp

```
1  int main() {
2      int a;
3      int b = -3;
4      int c = 12345;
5
6      int *p = &b;
7      return 0;
8  }
```

example2.cpp

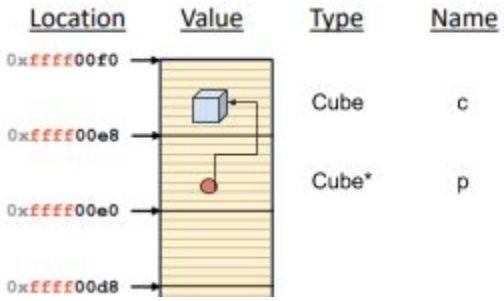
```
int main() {
   cs225::Cube c;
   cs225::Cube *p = &c;
   std::cout << "&c: " << &c <<
std::endl;
   std::cout << "&p: " << &p <<
std::endl;
   return 0;
}
```



Stack after each line of the code

CS 225 Spring 2019 :: TA Lecture Notes
1/18 Memory

By Wenjie



- (&p) = 0xffff00e0

- **Stack Frames**
  - Each function invocation gets a stack frame.
  - A stack frame is created whenever a function is called.
  - A stack frame is reclaimed when a function returns, and automatically marked free (not actually freed) . When memory is marked free and it can be overwritten. (We **never** want to return a pointer to the stack variable)

stackframe.cpp

```
1  int hello() {
2     int a = 100; //automatic
3  variables
4     return a;
5  }
```

```
6  int main() {
7    int a;
8    int b = -3;
9    int c = hello();
10   int d = 42;
11   return 0;
12 }
```