CS 180 - C++: Variable Types

Announcements
- Turn in HW now
- Program 1 posted
  - Checkpoint on Tuesday
- Lab is Friday, not Thursday
Types of Variable

1. Value
2. Reference
3. Pointer
2 Reference Variables

Syntax: `Point& c(a); // reference variable`

- `c` is created as an alias for `a`
- More like Python model, but can't be changed later

Example: `c = b;`
- Will not rebind `c` to point to `b`, but will change the value of `c` (and `a`).
Passing by reference:
Reference variables aren't usually needed in main program.
Instead, they are primarily used for passing to functions.

Ex:
```cpp
bool isOrigin(Point& pt) {
    return pt.getX() == 0 && pt.getY() == 0;
}
```

Advantage? - Changes persist
- Better speed, less space
If we want the speed of passing by reference but don't want our object mutated, use const.

```cpp
bool isOrigin(const Point& pt) {
    return pt.getX() == 0 && pt.getY() == 0;
}
```

Compiler will ensure that `pt` isn't modified.
Speeding up the Point class:

Original:
```cpp
double distance(Point other) const {
    // implementation
}
```

Faster:
```cpp
double distance(const Point& other) const {
    // implementation
}
```

Another:
```cpp
Point operator+(const Point& other) const {
    return Point(_x + other._x, _y + other._y);
}
```

Note: Return type is still value. Why?
```
Point created inside the function is destroyed at end of function.
Recall: Point output

```cpp
ostream& operator<<(ostream& out, Point p) {
    out << "(" << p.getX() << ", " << p.getY() << ")";  // display using form <x,y>
    return out;
}
```

Here, & is required because streams cannot be copied.

Note that we don't use const since we are changing the stream by adding data. If

```cpp
(cout << (c)) << endl;
```
3 Pointer variables

Syntax: 

```c
Point *d;  // d is a pointer variable
```

d is created as a variable that stores a memory address.
So:

```c
Point *d;  // d is a pointer variable

d = &b;

d = &c;
```

<table>
<thead>
<tr>
<th>d : Point*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xbfff1234</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b : Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 5.0</td>
</tr>
<tr>
<td>y = 7.0</td>
</tr>
</tbody>
</table>

But d is **not** a Point! can't say `d = b`
can't say `d.getx();`
Using pointer variables
2 options:

\[(\*d).get\ Y();\]

\[d \rightarrow get\ Y();\]
Passing by Pointer

Point *pt = NULL

bool isOrigin(Point *pt) {
    return pt->getX() == 0 && pt->getY() == 0;
}

This is similar to passing by reference but allows you to also pass a null pointer.
c = 5

(Python)

d = e

c = 6

d = 9

In C++, programmer is responsible for garbage collection.
More on Classes:

Destructors:

If your class opens files or allocates memory, then can't just use delete. Must create a destructor:

~ClassName() - no arguments, no return type

~Point()  // garbage collection - automatic in Python, not in C++

example with CreditCard
Copy Constructors:

Previously:
Point a;
Point b(a);

Consider the following:
Vector a (100);
Vector b (a);
What does this do?

copies each element of a to b.

\[ \text{int } \rightarrow \text{ vectSize } = \text{ a. vectSize } ; \]
\[ \text{array } \rightarrow \text{ the Vect } = \text{ a. the Vect } ; \]

\begin{center}
\begin{tabular}{|c|}
\hline
\text{a: Vect} \\
\text{vectSize} = 100 \\
\text{the Vect} \\
\hline
\end{tabular}
\end{center}
\begin{center}
\begin{tabular}{|c|}
\hline
\text{b: Vect} \\
\text{vectSize} = 100 \\
\text{the Vect} \\
\hline
\end{tabular}
\end{center}

Shallow copy
To fix, write our own copy constructor:

```cpp
Vect b(a);
```

// copy constructor
Vect (const Vect &a) {
    VectSize = a.VectSize;  // copy size
    for (int i = 0; i < VectSize; i++) {  
        theVect[i] = a.theVect[i];
    }
}

new is used for arrays
Another problem:

```
Vect a (100); 
Vect c; 
c = a.j;
```

What does this do? Shallow copy by default, copies each parameter.
```
c vectSize = a. vectSize; 
c. the Vect = a. the Vect.j;
```

Write operator = to make deep copy of data.
Enum: user defined types

```java
enum Color { RED, BLUE, GREEN }
```

```
0 1 2
```

```java
Color sky = BLUE;
Color grass = GREEN;
```

Convention: write in all capital letters