Variables, Types, Input/Output

Thomas Merkh
tmerkh55@gmail.com

Section 2, Week 2

April 7, 2020
Overview

1. Variables

2. Types

3. Input and Output

4. Can you decide what will be output?
1 Variables

2 Types

3 Input and Output

4 Can you decide what will be output?
Consider the example code

```cpp
#include <iostream>
using namespace std;

int main() {
    int x;
    int y = 1;
    x = y;
    cout << x << endl;
}
```

Here, you see several C++ language statements. In particular, you see `int x;` and `int y = 1;`. Here, x and y are called variables.

- `int x;` is “defining and declaring” variable x to the program
- `int y = 1;` is additionally “assigning” variable y the value 1
A word on the language being used:

- **declaring** a variable is to make the object known to the program. For example, `extern int x;`

- **defining** a variable is to allot space for the object in memory. For example, `int x;`

- **assigning** a variable is to provide the value that the variable takes, and is stored at the memory address of the variable. For example, `x = 3;`

- **initializing** a variable is generally referring to a variables initial assignment, which is sometimes chosen by default (discussed more later). For example, `int x = 3;`

Note: If you are using a compiled language, all entities must be declared before they are used!
There are certain rules to naming and declaring variables:

- A variable must have a *type* associated with it (more on this next)
- Upper and lower case letters are different, so $X$ and $x$ are not the same variables
- Although you *can* start a variable with `_` or `__`, but don’t! They can get confused with reserved pre-processor variables
- Variables can not start with a number, but can contain numbers after the first character

Note: For a list of reserved tokens that can’t be used as variable names: http://www.cplusplus.com/doc/tutorial/variables/
Variables

Types

Input and Output

Can you decide what will be output?
c++ is a called a “strongly typed language”, meaning you must state the “type” when declaring a variable. An abbreviated list of c++ types:

- **bool** - true or false, 1 or 0
- **char** - a variable which holds a single character or literal, such as ‘k’
- **int** - holds integer values
- **long** - holds integer values
- **float** - holds decimal values
- **double** - holds decimal values

And then there are more complicated “compound types”,
- **unsigned int** - an integer which can only take positive values
- **const double** - a double type which can not be reassigned

And these lists could keep going for many more slides...
c++ is a called a “strongly typed language”, meaning you must state the “type” when declaring a variable. An abbreviated list of c++ types:

- **bool** - true or false, 1 or 0
- **char** - a variable which holds a single character or literal, such as ‘k’
- **int** - holds integer values
- **long** - holds integer values
- **float** - holds decimal values
- **double** - holds decimal values

and then there are more complicated “compound types”,

- **unsigned int** - an integer which can only take positive values
- **const double** - a double type which can not be reassigned

and these lists could keep going for many more slides...
Warnings about Types

- Some compilers provide a default initialization (i.e. setting doubles to zero if not initialized), never use this if your compiler does this as your program won’t translate to other machines.
- The `long` type can hold larger integers than the `int` type, sometimes (sometimes the same).
- The `double` type can hold larger decimals than the `float` type, always(?)
  - This is because built-in data types have a specific number of bytes allocated to store their values. This can differ between computers.

Why care? Think of storing a 1000 by 1000 pixel image. That is 1 million values at least, and storing 1 million numbers should be done with care.
Each built-in type has properties associated with it and operators which can act on the type. For example, the computer must somehow know what the plus sign means in the cases:

```cpp
cout << 3 + 4 << endl;
>> 7

cout << 'a' + 'b' << endl;
>> 'ab'
```

Additionally, some details like numerical precision are being hidden here. There exist limits to the smallest number larger than 1 which a computer can represent with finite memory, and these considerations do sometimes come into play when coding. Buzzword: Machine epsilon.
The auto type is nice, but can be abused. In principle, declaring a variable using auto gives the compiler freedom to decide which type the variable needs to be, based on context. Consider

```c
int main() {
    int foo = 0;
    auto bar = foo;
}
```

Here, `bar` will be an integer type. The proper usage of this will become clear over the quarter. If this is used too heavily, debugging larger and more complex codes becomes nearly impossible.
1 Variables

2 Types

3 Input and Output

4 Can you decide what will be output?
My teacher once said: “If I can’t print out `<something>`, then I can’t even begin to code this”. I used to think I could complete programs without outputting anything until the end - don’t be like me, print everything, it will save you time.

The minimal structure of a c++ program you need in order to output (and input) to the computer’s terminal console is:

```cpp
#include <iostream>
using namespace std;

int main()
{
    return 0;
}
```
Let break this down:

```cpp
#include <iostream>
using namespace std;

int main()
{
    return 0;
}
```

- `using namespace std;` indicates we are using functions from the `standard` namespace, which holds many common built-in C++ routines like `cout`.
- `#include <iostream>` is a preprocessor directive, meaning that the contents of the iostream library could be used in this program.

Now, we could write statements like:

```cpp
std::cout << "Hello Earth" << std::endl;
```

or if we choose to not use `namespace std`, then:

```cpp
cout << "Hello Earth" << endl;
```
Often, users wish to interact with programs, leaving the question of how can a user input a numeric value into the program?

The answer is `cin`. Take for example:

```c++
#include <iostream>
using namespace std;

int main()
{
    int N;
    cout << "What is 7777777 mod 7?" << endl;
    cin >> N;
    cout << "The user input: " << N << endl;
    return 0;
}
```
1 Variables

2 Types

3 Input and Output

4 Can you decide what will be output?
What do you think the output will be?

```c
int main() {
    int x;
    int y = 0;
    x = 5;
    x = y;
    y = y - 1;
    cout << x << " " << y << endl;
}
```

The output will be "0 and -1". This is a simple example for now, but things will get much more complicated soon.

Thomas Merkh tmerkh55@gmail.com
April 7, 2020
What do you think the output will be?

```cpp
int main() {
    int x;
    int y = 0;
    x = 5;
    x = y;
    y = y - 1;
    cout << x << ' ' ' ' << y << endl;
}
```

The output will be “0 and -1”. This is a simple example for now, but things will get much more complicated soon.
Next discussion, we will discuss strings, math libraries, and type casting.

Any Questions?

Office Hours: Wednesday from 12-1:30pm.