19CSE201 :Advanced Programming

Lecture 3
Variables, Data types & Operators!

By
Ritwik M
Assistant Professor(SrGr)
Dept. Of Computer Science & Engg

Based on material from The Professor's C++Course by Linda W. Friedman and Stroustrup B. Programming: principles and practice using C++. Second edition,Addison Wesley; 2014.

A Quick Recap

- Imperative Paradigms
- Declarative Paradigms
- Programming language as a problem-solving tool
- Why have many Programming Languages
- Anatomy of C++
- I/O Streams

Variables

Variables (objects) must be declared as a certain type,
e.g., int, float, char, ...

Variables

This declaration may appear anywhere in the program, as long as it appears before the variable is first used.

The declaration creates the object.

For the declaration `int n;`

The name of the object is n, the type (or class) is int. The object does not yet have a value.

```
n = 66;    //now it has a value
```

```
int n=66; //This declaration also gives a value
```

More about variables

- A variable (object) has:
  - A name
  - A location in main memory.
    - This is an address in primary memory where the value of the object is stored while the program is executing.
  - A type (class).
    - A class defines the way the data item looks (e.g., char or int), how much space it takes up in memory, and how it operates.
  - A value.
    - It may have a value stored in the named location.
    - There are 3 ways to give a variable a value:
      - Give it an initial value at the declaration: `int n=66;`
      - Assign it a value using the assignment (=) operator: `n=66;`
      - Read the value in from an input device such as the keyboard: `cin >> n;`

Datatypes

- It is a template for how a particular set of values is represented in memory and what operations can be performed on those values.

- In C++ a type is the same as a class.

More on Datatypes

- Predefined data types
  - int, float, char, bool, double, etc..
- System-defined types
  - Part of the C++ class libraries.
  - Not part of the original C++ language definition but added when the compiler is written
  - cin, cout, string, etc..
- User-defined types
  - enum, type, class

Declarations

Declarations inform the compiler that it will need to set aside space in memory to hold an object of a particular type (class) with a particular name.

Types of Declarations

- Constant declarations
  - Used to associate meaningful names with constants
    - These will never change throughout the execution of the program.
  - One convention is to use all uppercase letters for constant identifiers.
  - Example:
    - `const float PI=3.14159;`
    - `const float METERS_TO_YARDS=1.196;`

Types of Declarations Cont.

- Variable declarations:
  - Used to associate identifiers of a given type with memory cells used to store values of this type.
    - The values stored in the data cells are changeable.
  - Example
    - `char letter;`
    - `char letter1, letter2;`
    - `float x, y;`

Types of Declarations Cont.

- Object declarations
  - Like variables, these are used to associate identifiers of a given type with memory cells used to store values of this type.
    - The values stored in the data cells are changeable.
    - We use some system-defined classes in the standard C++ class libraries.
    - A class is equivalent to a type; variables can store data values and are called objects.
  - Example
    - `ofstream cprn ("printfile.txt");`

Have you studied something similar to this before?

Typecasting

- We can convert one datatype to another

- Explicit Typecasting
  - Example:

```
char c='A';
cout<<int(c);
```

----------> outputs 65

- Implicit Typecasting
  - Example:

```
int a=20; float b=3.0;
cout<<a/b;
```

----------> outputs 6.66667

Variable Limits

Try this on HPOJ

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    cout << "minimum char = " << CHAR_MIN << endl;
    cout << "maximum char = " << CHAR_MAX << endl;
    cout << "minimum short = " << SHRT_MIN << endl;
    cout << "maximum short = " << SHRT_MAX << endl;
    cout << "minimum int = " << INT_MIN << endl;
    cout << "maximum int = " << INT_MAX << endl;
    cout << "maximum long = " << LONG_MAX << endl;
    cout << "maximum unsigned = " << UINT_MAX << endl;
    cout << "maximum unsigned short = " << USHRT_MAX << endl;
    cout << "maximum unsigned long = " << ULONG_MAX << endl;
    cout << endl << endl << endl;
    return 0;
}
```

Overflow

- Happens if we try to store a value that is too large for the data type to handle
- This is a run-time error.
- Example:

```
#include<bits/stdc++.h>
using namespace std;
int main(){
        short n= SHRT_MAX - 1;
        cout << n++ << endl;
        cout << n++ << endl;
        cout << n++ << endl;
        cout << n++ << endl;
        return 0;
   }
```

What did you observe?

Operators

- Arithmetic Operators
  - Binary Operators
  - *, /, %, +, -
  - Order of precedence Left to Right
- Increment / Decrement Operators
  - Unary Operators
  - ++, --
  - Pre vs post increment/decrement
- Note: The degree of an operator refers to the number of operands it takes.

The Math Library

- Built in library for mathematical operations
- use `#include<math.h>`

function	what it does	returned value
abs(a)	absolute value of a	same data type as argument
pow(a1,a2)	a1 raised to the power of a2	data type of argument a1
sqrt(a)	square root of a	same data type as argument
sin(a)	sine of a (a in radians)	double
cos(a)	cosine	double
tan(a)	tangent	double
log(a)	natural logarithm of a	double
log10(a)	base 10 log of a	double
exp(a)	e raised to the power of a	double

A Short Note on Errors

- Syntax errors – compile-time errors
  - These errors are picked up by the compiler and we will usually get error messages about them. Syntax errors result from using the language incorrectly.

- Logic errors – run-time errors
  - These errors are generally not flagged by the system. We find out about them by checking the output to see whether it is correct.

Quick Summary

- Variables
- Datatypes
- Declaration
- Type casting
- Overflow
- Operators
- Errors

Up Next

Selection, Repetition and Functions