CS 150 Introduction to Computer Science I

Data Types

Today

- Last we covered
  - **`main`** function
  - **`cout`** object
  - How data that is used by a program can be declared and stored
- Today we will
  - Investigate the various types of data that C++ can handle

Declaration Statements

- Examples of declaration statements

```
const double PI = 3.14;
const double RADIUS = 5.4;
double area;
double circ;
```

- With the above statements we are declaring four things
  - **`PI`** to store the value of Pi that never changes
  - **`RADIUS`** to store the value of radius that never changes
  - **`area`** to store the area of the circle
  - **`circ`** to store the circumference of the circle

Declaration Statements

- Variable declarations
  - Allocate space for data to be used in the program
  - The data *can* be changed

```
double area;
double circ;
```

- Constant declaration
  - Allocate space for data that *cannot* be changed

```
const double PI = 3.14;
const double RADIUS = 5.4;
```


Variable Declaration

- Variables are declared by stating
  - Type of data (data type)
  - Name to identify the variable (identifier)
  - Semicolon (;)

```
data-type identifier;
double area;
```


Variable Declaration

- If there is more than one variable of a single data type then you
  - State the data type
  - List the variable identifiers (names) separated by commas
  - Semicolon

```
data-type identifier1, identifier2;
double area, circ;
```

Identifiers

- Programmer-defined names that represent some element of a program
- C++ does place limits on what names you can call your variables
- Rules
  1. Identifiers must begin with a letter or an underscore
  2. Identifiers must consist of letters, numbers and underscore, nothing else
  3. Identifiers cannot be a *reserved keyword*

Reserved Keywords

- These are words that are reserved by C++ to implement various features
- Examples of keywords that we have seen so far are **`int, double, const, return`**
- A list of C++ keywords can be found on page 45 of your textbook

Identifiers

- Identifiers are case sensitive
  - `int num1;`
  - `int Num1;`
  - `num1` and `Num1` are different variables
- You should always try to use meaningful variable names
- If you have a variable that represents the width, then call it **`width`** not **`w`**

Identifiers

- Q 4.1 Which of the following declarations are invalid and why?
  a. `char Letter1;`
  b. `char 1letter;`
  c. `double inches, kms;`
  d. `double inches*num;`
  e. `int joe's;`
  f. `Int cent_per_inch;`
  g. `double two-dimensional;`
  h. `char hello;`
  i. `int return;`
  j. `size int;`

Data types

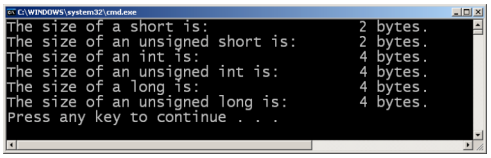
- Data types
  - C++ can store many different types of data
  - A data type also defines what operations can be performed on data of that type
  - We will be looking at
    - Integer numbers
    - Characters
    - Strings
    - Floating-point numbers
    - Booleans

Integers

- The main integer data type is `int`
- The `int` data type is used to store integer numbers, both positive and negative
- `int`s are finite (why?), i.e. they have a limited range that is implementation dependent
- Examples of `int`s are: 123, -23, 0, 2352
- An `int` without a sign (+ or -) is assumed to be positive
- 2,353 is not an `int`, 2353 is an `int`
- What operations can be performed on integers?

Integer Data Types

- There are five integer data types, each with a different range and a different size



- Range of data types is listed on page 48

char

- The `char` data type is used to store single characters (letters, digits, special characters)
- `chars` are usually 1-byte long
- Characters are stored as integers
- The most common method for encoding characters is ASCII
- Character constants are enclosed in single quotes
- Examples of character constants are: `'A', 'a', '*', '2', '$'`

Program 4.1

```
#include "stdafx.h"
#include <iostream>

using namespace std;

int main()
{
   char letter;

   letter = 65;
   cout << letter << endl;
   letter = 66;
   cout << letter << endl;
   return 0;
}
```


Program 4.2

```
#include "stdafx.h"
#include <iostream>

using namespace std;

int main()
{
   char letter;

   letter = 'A';
   cout << letter << endl;
   letter = 'B';
   cout << letter << endl;
   return 0;
}
```


char

- Character constants can only hold a single character
- String constants are used to store a series of characters
- To indicate the end of a string, a null terminator is used

Questions

- Q 4.2 How are the character 'A' and the string constant "A" stored in memory?
- Q 4.3 Is the escape character \n a character or a string?
- Q 4.4 How do we declare a `char` variable and assign it a value?

`string` Class

- **`string`** is the data type used to store more than one character
- Not built into C++ but provided by standard C++
- Need to include the preprocessor directive
  - `#include <string>`

`string` Questions

- Q 4.5 How do we declare a variable of type string?
- Q 4.6 How do we assign a value to the variable?
- Q 4.7 How do we output a string constant and a string variable? What is output?

Floating-Point Data Types

- The **`float`**, **`double`**, **`and`** **`long double`** data types are used to store floating-point numbers, both positive and negative
- Floating-point numbers can contain fractional parts
- Computers store floating-point numbers in a manner similar to scientific notation
- Computers represent floating-point numbers using E notation

Floating-Point Data Types

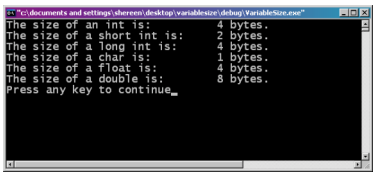
- **`float`**, **`double`**, **`and`** **`long double`** are finite
- Examples of floating-point numbers are: 1.0, -2.3, -.3, 12E5, -1E-2, 1.4e+8
- 2,353.99 is not a **`double`**, 2353.99 is a **`double`**

How to Choose a Data Type

- Ask yourself the following questions
  - What are the largest and smallest numbers that may be stored?
  - How much memory does the variable use?
  - Is the variable signed (positive and negative)?
  - How many decimal places of precision does the variable need?

Variable Sizes

- On my machine the sizes are



Variable Size Program

```
#include "stdafx.h"
#include <iostream>

using namespace std;

int main()
{
  cout << "The size of an int is:\t\t"    << sizeof(int)    << " bytes.\n";
  cout << "The size of a short int is:\t" << sizeof(short)  << " bytes.\n";
  cout << "The size of a long int is:\t"  << sizeof(long)   << " bytes.\n";
  cout << "The size of a char is:\t\t"    << sizeof(char)   << " bytes.\n";
  cout << "The size of a float is:\t\t"   << sizeof(float)  << " bytes.\n";
  cout << "The size of a double is:\t"    << sizeof(double) << " bytes.\n";

  return 0;
}
```


Variable Ranges

Type	Size	Values
`int`	4 bytes	–2,147,483,648 *to* 2,147,483,647
`short int`	2 bytes	–32,768 *to* 32,767
`long int`	4 bytes	–2,147,483,648 *to* 2,147,483,647
`unsigned int`	4 bytes	0 *to* 4,294,967,295
`char`	1 byte	256 character values
`float`	4 bytes	1.2e–38 *to* 3.4e38
`double`	8 bytes	2.2e–308 *to* 1.8e308

Summary

- In today's lecture we covered
  - Identifiers
  - Data types
  - How data that is used by a program can be declared and stored
- We have covered p. 45 - 63 of your textbook