int Data Type, Arithmetic Operators, and Variables

CSE100

Sections 2.7, 2.15, 3.2, 2.5-2.6

cout Problems

1. What is the output of the following code segment:

   ```
   cout << "I am the \"incredible\"";
   cout << "computing\n\tmachine";
   cout << "\nand I will \n\namaze\n";
   cout << "\tyou.\n";
   ```

2. Find and fix the errors:

   ```
   Cout << "red /n" << "blue \ n" << "yellow"  "endl" <<
   green  << \n;
   ```

3. Write a complete program that displays your name on the first line, your street address on the second, and city, state, ZIP on the third.

cout Problems

1. What is the output of the following code segment:

```
cout << "I am the \"incredible\"";
cout << "computing\n\tmachine";
cout << "\nand I will \n\namaze\n";
cout << "\tyou.\n";
```

```
I am the "incredible"computing
        machine
and I will

amaze
        you.
```

cout Problems

2. Find and fix the errors:

```
Cout << "red /n" << "blue \ n" << "yellow"  "endl" <<
green  << \n;
```

```
cout << "red \n" << "blue \n" << "yellow"  << endl << "green"
<< "\n";
```

Data Type Intro and int Data Type

Data Types in C++

- Data Type: set of values together with a set of operations is called a data type
- C++ data can be classified into four categories:
  - **Simple data type (Initially we will focus on simple data types)**
  - **Structured data type (In this class we will learn arrays)**
  - **Abstract Data Types (In this class we will learn classes)**
  - **Pointers (Time permitting)**

Simple Data Types

- Two broad categories of simple data types
  - numeric
    - Integral: integers (numbers without a decimal)
    - Floating-point: decimal numbers
  - character

int Data Type

- Can take any integral value between -2147483648 and 2147483647
- Stored in 4 bytes of memory
- Stored in memory as a binary digit
  - Example in 4 bits
    - 0 = 0000
    - 5 = 0101
    - 7 = 0111
    - -1 = 1111
    - -5 = 1011
    - -8 = 1000

int Data Type

- Examples:

```
-6728
0
78
```

- Positive integers do not have to have a + sign in front of them
- No commas are used within an integer
  - Commas are used for separating items in a list
- int is a keyword

int Constants

Any integer that is not in quotes (part of a string) that is part of a program.

```
//Simple program showing int constants
#include <iostream>
using namespace std;

int main()
{
    cout << "The sum of " << 5 << " and 3 = " << 8;
    return 0;
}
```

Operators

- Used to perform operations on data, called the *operands*
- Many types of operators work with ints
  - Arithmetic: `+, -, *, /, %`
  - Assignment: `=`
  - Stream: <<
- Between ints, / is integer division results in the quotient of the result. Truncates answer to int:
  - 17/5 => 3
  - 24/9 => 2
- % is the *modulus* operator. The result is the remainder of the division
  - 17%5 => 2 //Read 17 mod 5 is 2
  - 24%9 => 6
  - 15%5 => 0

Order of Operations

Operator	Associativity	Description
()	left to right	Group operations
+, - (unary)	right to left	negate a number
*,/,%	left to right	arithmetic
+, - (binary)	left to right	arithmetic
<<	left to right	stream insertion

Expressions

- An expression is programming statement that evaluates to a value
  - EX: 3 + 2 is an expression that evaluates to 5

- Some other examples showing order of precedence:
  - (-4 + 17) % 2*2 - 1 evaluates to 1
  - 8 + 12 * (6 - 2) evaluates to 56

Arithmetic Question

What's the result of the following:

27 / 4 - 2

5 + 3 * 4

6 + 17 % 3 - 2

12 / (6 - 10)

2 / 3 * 7 + 2 % 3

Identifiers and Variables

Identifiers

- Programmer-Defined Symbols
- Not part of the C++ language

- Used to represent various things
  - variables (memory locations), functions, etc.
- Example in program (shown in green):

```
int number1;
```

Forming Identifiers

- Cannot be keywords
- Must begin with a letter (a-z or A-Z) or underscore (_)
- Can be followed by letters, digits (0-9), and the underscore character (_)

- C++ is case sensitive
  - Name and name are different identifiers
  - payRate, payrate and PayRate are all different

Identifier Example

- Are the following valid or invalid (and why) identifiers:
  - first
  - 2nd
  - employee Salary
  - one+two
  - payRate
  - Hello!
  - conversion
  - _4teen

Variables

- A *variable* is a named location in main memory.
  - Called variable because the value stored at that location may change
- Variables consist of two parts:
  - A data type, so the computer knows how to store the data and the operations that can be performed on it.
  - An *identifier*, the name we give the variable

Variable Definition and Declaration

- *Variable Definition* - Assigning the variable a specific place in main memory.
  - Must be done before the variable can be used.
  - Done by specifying the data type followed by an identifier in a statement
    - In general: <data type> <identifier>;
    - Example: int number;
  - More than one variable can be defined in the same statement if identifiers are separated by commas
    - Example: int number1, number2;

Variable Initialization

- *Variable Initialization* - Assigning a value to the variable for the first time.
  - Some compilers will cause an error if the variable is used without initialization
  - Others will print out an unpredictable value
  - Example: int number1;

    number1 = 5;
  - Can be done on the same line as the definition
    - int number = 5;
    - More than one number can be initialized at once in a definition line:
      - int number1 = 5, myInt, num3 = 8;

= Operator

- Called the *Assignment Operator*
- Evaluates the expression on the right and stores it in the variable on the left
  - Always need to have a variable on the left
  - The expression on the right, must evaluate to a value of the variable's data type
  - Any previous value at that location is replaced

- Example:

```
1. int num;
2. num = 5;
3. num = num + 12;
4. int num2 = 8;
5. num = num2 + 5;
6. num = num + num2 * 2;
```

= Operator

- Example of WRONG use:
  - 4 + num2 = num;

- Can chain together assignments
  - Always operates from right to left
- Example:
  - int num1, num2;
  - num1 = num2 = 5*3;

Variable Output

- To output variables of simple data types, insert the variable name into the stream
  - Note the variable name does not go in quotes
    - What would happen if it did?
  - With variables number1, number2 and sum

```
cout << "\nThe sum of " << number1 << "
  and " << number2 << " is  " << sum <<
  endl;
```

- Be careful to use spaces in the strings around the insertion of the variables for readability.

Arithmetic Operators

- Work the same way as for int constants
- Need to always put the operator in between constants and variables
- Can use directly in output stream

Examples:

```
int x=2, y=4, z=5;
cout << 2*x+y;  // not 2x+y
cout << (x+y) / (z+2);  //What happens if missing ()
cout << x*y+z;   // not xy + z
```

Tips for Variables

- A variable MUST be defined before it's used
- Only define a variable once
- Names should be meaningful
  - If the variable is going to hold the length, should call it *length*, rather than l or x
  - Makes the program self-documenting
- By convention, variable names should start with a lowercase letter
- If the name is more than one word
  - Start the second word with a capital letter
    - int payRate;
  - OR separate the words with an underscore
    - int pay_rate;

Program Example: Variables

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


int main()
{
   int number1, number2 = 7;
  number1 = 5;
  cout<< "The value of number1 is " << number1 << endl;

  int sum = number1 + number2;

  cout << "\nThe sum of " << number1 << " and " << number2
  << " is " << sum << endl;

   return 0;
}
```

Book Problems

The following problems are for extra practice, but are not due.

Checkpoint questions on pages 41, 48 (2.10, 2.11, 2.14, 2.15 only), and 90 (3.9-3.11 only). (answers in Appendix C)

Review Questions page 69-72 : 9, 13, 16, 25A,B, 26B

Programming Challenges page 73: 1, 14

Next Time

- input with cin (3.1)
- Problem solving process (1.5, 1.6)
- More operators (3.6, 5.3)