CS118 – Programming Fundamentals

Lecture # 07
Saturday, September 14, 2019
FALL 2019
FAST – NUCES, Faisalabad Campus

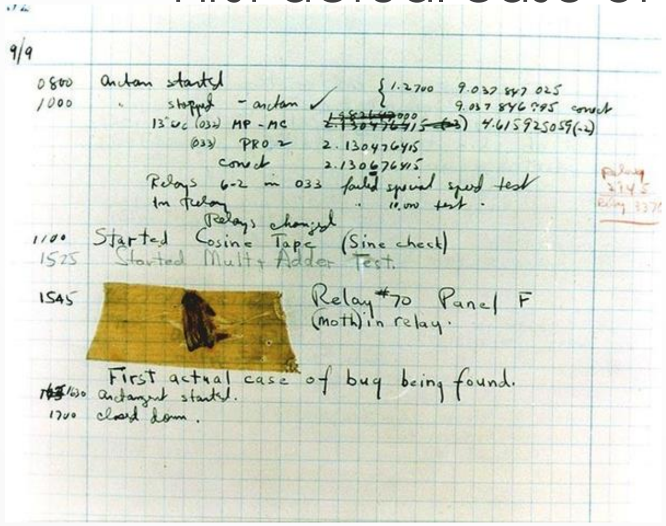
Zain Iqbal

Basic Components of C++ Program

Testing and Debugging

- Bug
  - A logical mistake in a program
- Debugging
  - Eliminating mistakes in programs
  - Term used when a moth caused a failed relay on the Harvard Mark 1 computer. Grace Hopper and other programmers taped the moth in logbook stating:

"First actual case of a bug being found."



9/9

0800 Arctan started

1000 " stopped - arctan ✓ {1.2700 9.037 847 025
9.037 846 995 conect

13" uc (032) MP - MC ~~2.130476415~~ (-3) 4.615925059(-2)

(033) PRO 2 2.130476415

conect 2.130676415

Relays 6-2 in 033 failed special speed test
In relay " 11.000 test.

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

1630 Arctangent started.

1700 closed down.

Relay 2145
Relay 3370

Program Errors

- **Syntax errors**
  - Violation of the grammar rules of the language
  - Discovered by the compiler
  - Error messages may not always show correct location of errors
- **Run-time errors**
  - Error conditions detected by the computer at run-time
- **Logic errors**
  - Errors in the program’s algorithm
  - Most difficult to diagnose
  - Computer does not recognize as an error

What makes a bad program?

- Writing Code **without detailed analysis and design**
- Repeating trial and error **without understanding the problem**
- Debugging the program line by line, statement by statement
- **Writing tricky and dirty programs**

Formatting Output

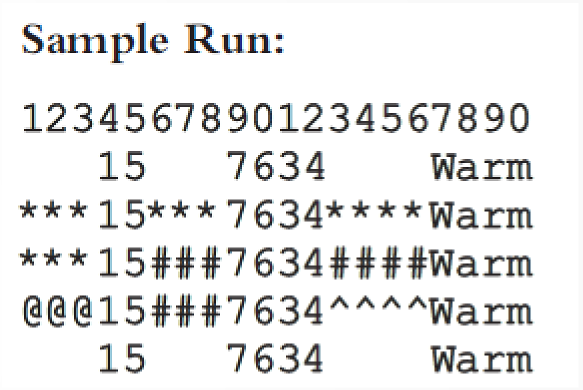
Manipulators

- A manipulator functions format the output to present it in more readable fashion

For example

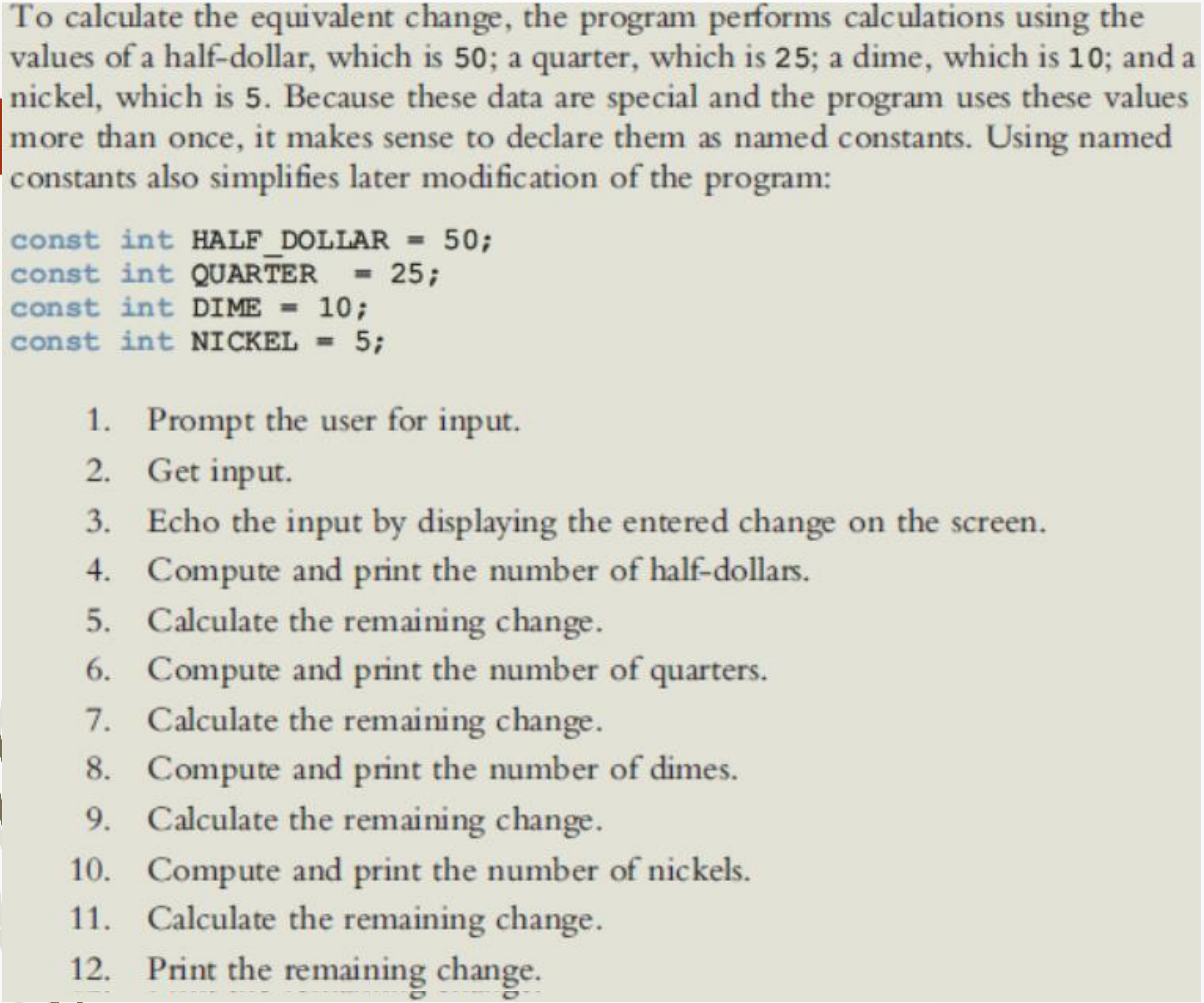
- **endl** -- New line
  - `cout << “Hello Dear…” << endl ;`
- **setw(…)** -- set width of output fields
  - `cout << setw(10) << “Hello”<<“\t |1o characters width” << endl ;`
- **setfill(…)** -- specifies fill character
  - `cout << setw(10) << setfill(‘*’) << “Hello”<<“\t |find difference in format of output” << endl ;`
- **setprecision(…)** -- specifies number of decimals for floating point : e.g. `setprecision(2) ….`

```cpp
//Example setfile
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     int x = 15;                    //Line 1
     int y = 7643 ;                 //Line 2

     cout << "12345678901234567890" << endl;        //Line 3
     cout << setw(5) << x << setw(7) << y
          << setw(8) <<"Warm" << endl ;             //Line 4

     cout << setfill('*');                          //Line 5
     cout << setw(5) << x << setw(7) << y
         << setw(8) << "Warm" << endl ;             //Line 6

     cout << setw(5) << x << setw(7) << setfill('#')
         << y << setw(8) << "Warm" << endl ;        //Line 7

     cout << setw(5) << setfill('@') << x
          << setw(7) << setfill('#') << y
          << setw(8) << setfill('^') << "Warm"
          << endl ;                                 //Line 8

     cout << setfill(' ') ;                         //Line 9
     cout << setw(5) << x << setw(7) << y
          << setw(8) << "Warm" << endl;             //Line 10
     return 0;
}
```



Sample Run:

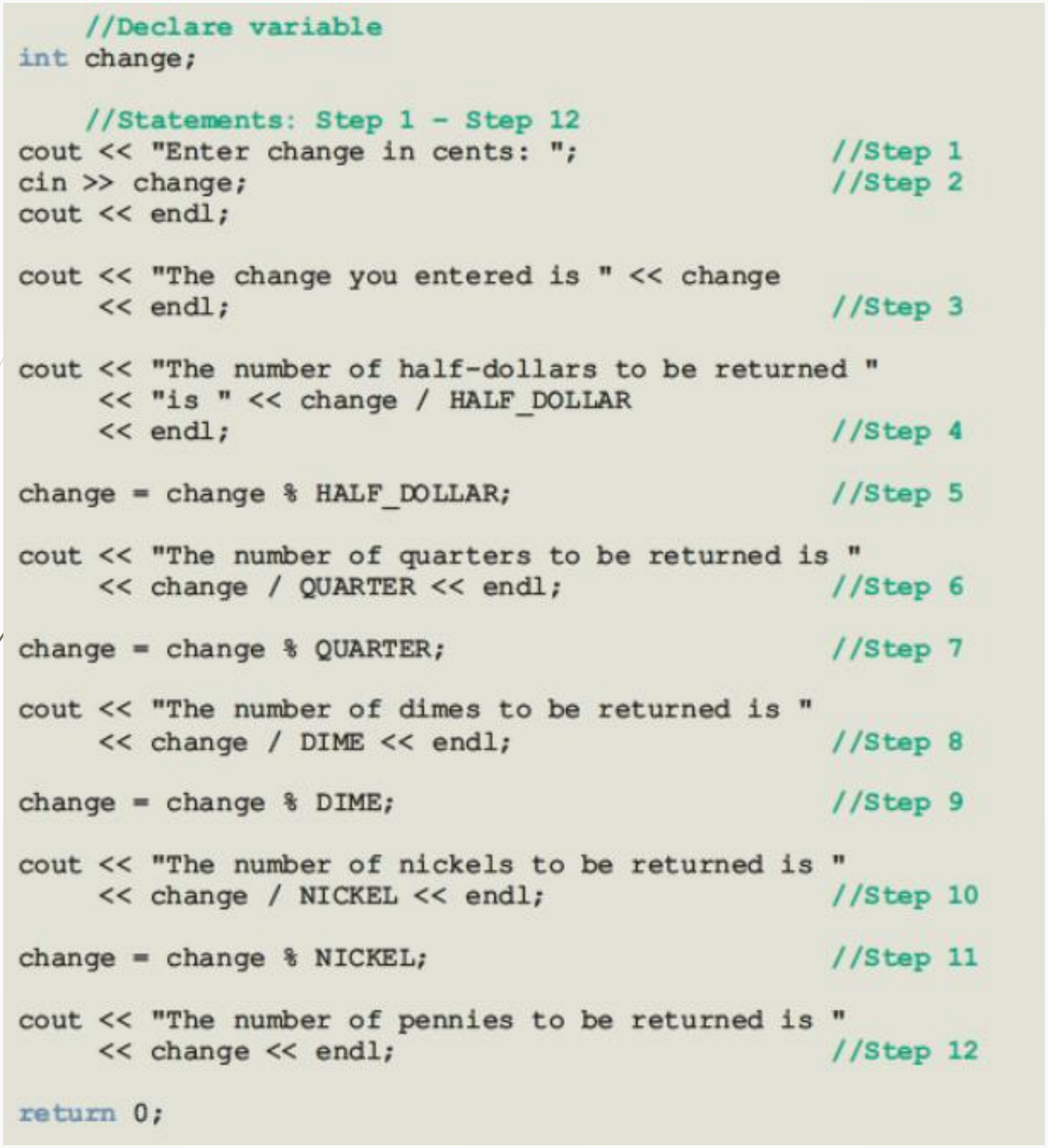
```
12345678901234567890
   15   7634    Warm
***15***7634****Warm
***15###7634####Warm
@@@15###7634^^^^Warm
   15   7634    Warm
```



To calculate the equivalent change, the program performs calculations using the values of a half-dollar, which is 50; a quarter, which is 25; a dime, which is 10; and a nickel, which is 5. Because these data are special and the program uses these values more than once, it makes sense to declare them as named constants. Using named constants also simplifies later modification of the program:

```
const int HALF_DOLLAR = 50;
const int QUARTER  = 25;
const int DIME = 10;
const int NICKEL = 5;
```

1. Prompt the user for input.
2. Get input.
3. Echo the input by displaying the entered change on the screen.
4. Compute and print the number of half-dollars.
5. Calculate the remaining change.
6. Compute and print the number of quarters.
7. Calculate the remaining change.
8. Compute and print the number of dimes.
9. Calculate the remaining change.
10. Compute and print the number of nickels.
11. Calculate the remaining change.
12. Print the remaining change.



```
    //Declare variable
int change;

    //Statements: Step 1 - Step 12
cout << "Enter change in cents: ";                     //Step 1
cin >> change;                                          //Step 2
cout << endl;

cout << "The change you entered is " << change
     << endl;                                           //Step 3

cout << "The number of half-dollars to be returned "
     << "is " << change / HALF_DOLLAR
     << endl;                                           //Step 4

change = change % HALF_DOLLAR;                          //Step 5

cout << "The number of quarters to be returned is "
     << change / QUARTER << endl;                       //Step 6

change = change % QUARTER;                              //Step 7

cout << "The number of dimes to be returned is "
     << change / DIME << endl;                          //Step 8

change = change % DIME;                                 //Step 9

cout << "The number of nickels to be returned is "
     << change / NICKEL << endl;                        //Step 10

change = change % NICKEL;                               //Step 11

cout << "The number of pennies to be returned is "
     << change << endl;                                 //Step 12

return 0;
```

Fahrenheit to celsius

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

int main()
{
	int fahrenheit, celsius ;
	cout << " Enter temperature in Fahrenheit : " ;
	cin >> fahrenheit ;
	cout << endl;

	celsius = 5.0 / 9.0 * (fahrenheit - 32) ;

	cout << fahrenheit << " degree F =  "
		<< celsius << " degree C" << endl;
	return 0;
}
```

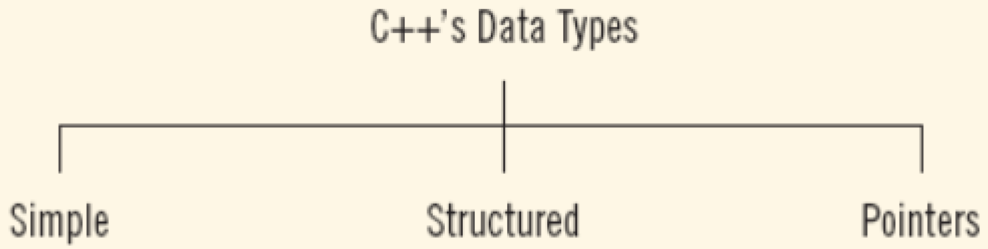
Objectives

In this lecture, you will:

- Explore simple data types
- Discover how to use arithmetic operators
- Examine how a program evaluates arithmetic expressions
- Learn what an assignment statement is and what it does
- Become familiar with the string data type
- Become familiar with the use of increment and decrement operators
- Explore how to properly structure a program, including using comments to document a program
- Learn how to write a C++ program

Data Types

- **Data type:** Set of values together with a set of operations
- C++ data types fall into three categories:



Simple Data Types

Three categories of simple data

- **Integral:** integers (numbers without a decimal)
- **Floating-point:** decimal numbers
- **Enumeration type:** user-defined data type

Simple Data Types (cont'd.)

Integral data types are further classified into nine categories:

- char, short, int, long, bool
- unsigned char, unsigned short, unsigned int, unsigned long

Simple Data Types (cont'd.)

- Different compilers may allow different ranges of values

Data Type	Values	Storage (in Bytes)
int	-2147483648 to 2147483647	4
bool	true and false	1
char	-128 to 127	1

int **Data Type**

Examples:

-6728

0

78

+763

- Positive integers do not need a + sign
- **No commas** are used within an integer [76,385]
  - Commas are used for separating items in a list

bool **Data Type**

- bool type
  - Two values: true and false
- Manipulate logical (Boolean) expressions
- true and false
  - Logical values
- bool, true, and false
  - Reserved words

char **Data Type**

- The smallest integral data type
- Used for characters: letters, digits, and special symbols
- Each character is enclosed in single quotes
  - 'A', 'a', '0', '*', '+', '$', '&'
- A blank space is a character
  - Written ' ', with a space left between the single quotes
  - ‘abc’ and ‘!=‘ are not char

Floating-Point Data Types

- You may be familiar with scientific notation. For example:
  - 43872918 = 4.3872918 * 10^7 {10 to the power of seven}
  - .0000265 = 2.65 * 10^{-5} {10 to the power of minus five}
  - 47.9832 = 4.79832 * 10^1 {10 to the power of one}

Floating-Point Data Types

- C++ uses scientific notation to represent real numbers (floating-point notation)

Real Number	C++ Floating-Point Notation
75.924	7.592400E1
0.18	1.800000E-1
0.0000453	4.530000E-5
-1.482	-1.482000E0
7800.0	7.800000E3

Floating-Point Data Types (cont'd.)

- **float:** represents any real number
  - Range: -3.4E+38 to 3.4E+38 (four bytes)
- **double:** represents any real number
  - Range: -1.7E+308 to 1.7E+308 (eight bytes)

Floating-Point Data Types (cont'd.)

- Maximum number of significant digits (decimal places) for float values is 6 or 7
- Maximum number of significant digits for double is 15
- **Precision:** Maximum number of significant digits
  - float values are called single precision
  - double values are called double precision

Data type and variables

- When we declare a variable we not only specify the variable we also specify what type of data a variable can store. A syntax rule to declare a variable

```
datatype identifier;
```

- For example consider the following examples:

```
int counter;
double interestRate;
char grade;
```

Datatypes and their ranges

Type	Width	Common Range
char	8	-128 to 127
unsigned char	8	0 to 255
int	32	–2,147,483,648 to 2,147,483,647
unsigned int	32	0 to 4,294,967,295
short int	16	-32768 to 32767
unsigned short int	16	0 to 65535
long int	64	-2,147,483,648 to 2,147,483,647
unsigned long int	64	0 to 4,294,967,295
float	32	3.4E-38 to 3.4E+38
double	64	1.7E-308 to 1.7E+308
long double	80	3.4E-4932 to 3.4E+4932