CS 102 Introduction to Programming Using C++

Chapter 3

Decisions

Homework

- Written homework
- R3.1 to 5, 7, 13, 14, 19, 20, 27, 30

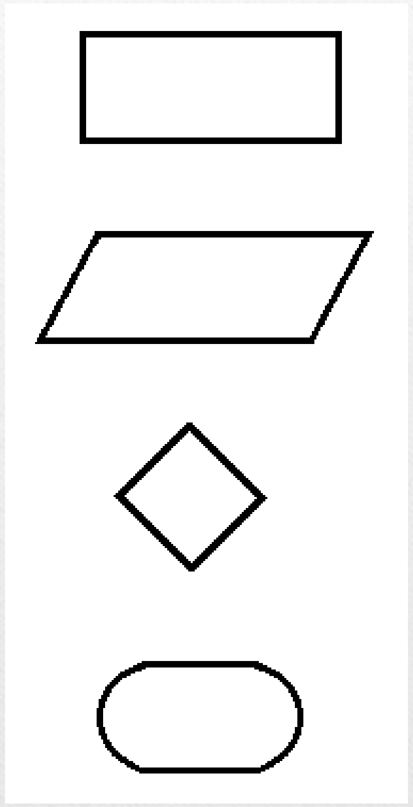
- Programming Assignments
- p. 118, P3.1, 3, 5, 6, 7, or 25
- Also, one of P3.16, 19, and 25

Structured Programming

- Programs can be diagrammed using flowcharts
- A flowchart gives a "picture" of a program
- Items in a flowchart are represented by shapes
- The shapes are connected by lines to show connections
    - The connections indicate the order in which the statements are executed
- Flowcharts will be explained later in the chapter
- We have come to use only three structures

Flow Charts

- A flow chart has many symbols
  - They are all connected by lines
  - The lines always flow down or to the right
- A rectangle indicates a task
  - It can be a single statement or several statements
- A non-rectangular parallelogram indicates input or output
  - For us, that's cin, cout
- A diamond indicates a decision
  - It should be followed by yes and no, or some decision
- Also, there are usually start and end symbols



The Three Structures of Structured Programming

- Sequence
  - This means executing the lines "in order"
- Decision
  - This means doing one thing out of several
  - A choice is consciously made
  - This choice is based on some kind of fact
  - Usually it's a comparison
- ??? (We will learn the third structure in Chapter 4)

The <u>if</u> Statement

- The <u>if</u> statement is a way to make a choice
- Here is an example

```
double service_charge = 0.00;
if (account_balance < 0.00)
    service_charge = 25.00;
```

- Notice the indenting
- Read the textbook's note on tabs on p.81

Adding Alternatives

- Let's make a slight change to the previous code fragment

```
double service_charge;
if (account_balance < 0.00)
    service_charge = 25.00;
else
    service_charge = 0.00;
```

Adding Braces

- Adding braces avoids problems
- Here is the previous example with braces

```
double service_charge;
if (account_balance < 0.00)
{
    service_charge = 25.00;
}
else
{
    service_charge = 0.00;
}
```

Ouch! Legal, But Usually a Logic Error

- The <u>if</u> part of the <u>if</u> statement itself should not get a semi-colon
- This is perfectly legal, but it usually is not intentional

```
if  (account_balance < 0.00);
        service_charge = 25.00;
else
        service_charge = 0.00;
```

- Can you spot the minor difference in appearance?

Avoid Duplicate Code

- There is an example on p.82 that talks about duplicate code
- For example

```
double service_charge;
if (account_balance < 0.00)
{
    service_charge = 25.00;
    total_charges = total_charges + service_charge;
}
else
{
    service_charge = 0.00;
    total_charges = total_charges + service_charge;
}
```

Remove Duplicate Code from the <u>if</u>

- This is better

```
double service_charge;
if (account_balance < 0.00)
{
    service_charge = 25.00;
}
else
{
    service_charge = 0.00;
}
total_charges = total_charges + service_charge;
```

Comparison Operators

- They are called relational operators
- They are

  < (less than)

  <= (less than or equal)

  > (greater than)

  >= (greater than or equal)

  == (equal—Be careful!)

  This is not the same as =

  != (not equal)

Comparing Floating Point Numbers

- Comparing floating point numbers
  - <, >, <=, >= are okay
  - We never ask if two floating point numbers are equal
    - Because of round-off errors, they could be really, really close
    - Effectively, they are equal
  - Instead, we check if they are very close
  - The test is

```
if (fabs (double1 – double2) < epsilon))
```

- This is partly context-dependent

Comparing Strings

- This is like comparing numbers
- Notice that “HI” and “Hi” are not equal
  - For two strings to be equal, they must match exactly
- if (string1 < string2)
  - means if string1 comes before string2 in the dictionary
  - Dictionary order is called lexicographic order

Multiple Alternatives

- We want to write a program to assign grades
- The scale is
  - 90-100 ->A
  - 80-89.999…->B
  - 70-79.999…->C
  - 60-69.999…->D
  - <60 ->F

The Code

```
if (grade >= 90)
{
    cout << “A”;
}
else if (grade >= 80)
{
    cout << “B”;
}
else if (grade >= 70)
{
     cout << “C”;
}
else if (grade >= 60)
{
    cout << “D”;
}
else
{
    cout << “F”;
}
```

Analyzing the <u>if</u>

- Notice that the <u>if</u> is combined with the <u>else</u> on a single line
    - This increases readability
- Notice how the code works
    - Each <u>if</u> "cuts off" a part of the grade range

The Last Choice in a Multi-way if

- Notice that the last else is not an else if
  - The last else is a "catch-all"
    - It catches anything that is still left
  - If the last else were an else if, it's possible that all cases of the if would be skipped
  - Good programming practice (Read this as "something that increases readability") is to comment the last else

```
else  /*  if (grade >= 0   &&   grade < 60)  */
```

Nested <u>ifs</u>

- You can put one <u>if</u> inside another
- For example
- You have several shirts
  - They're all light colored, except there is one dark blue shirt
- On Tuesday, you always wear your dark blue shirt
  - However, if it's raining, you wear your bright yellow shirt, because dark colors are too depressing
- On other days, you wear whatever you can find

Coding the <u>if</u>

- We could write this pseudocode

```
if (today == "Tuesday")
    if (weather == "raining")
        Wear yellow shirt
    else
        Wear blue shirt
else
    Wear any shirt
```

- Braces deleted due to lack of space

Hand Tracing

- This is also called desk checking
- It is when you trace a program without using a computer
- In class, I often hand trace programs
- You should develop this habit
- Desk check your programs several times before running them

Shipping Charges

- We will write a program to calculate shipping charges for an international company
- The company ships to several international destinations
- Right now, Hawaii is the only US destination , but the company plans on expanding to other states
- Shipping charges are as follows:
  - Hawaii $10.00
  - International cities $20.00

Problem: A Dangling Else

- The code:

```
if  (country == “USA”)
   if  (state == “HI”)    // HI is the code for Hawaii
      shipping_charge = 10.00;
else
   shipping_charge = 20.00;
```

  - Remember, indentation is only for people
  - The second else matches the LATEST if

The Solution

- The code:

```
if (country == “USA”)
{
    if (state == “HI”)
    {
        shipping_charge = 10.00;
    }
}
else
{
   shipping_charge = 20.00;
}
```

The boolean Data Type

- C++ has a boolean data type
- A boolean variable can store <u>true</u> or <u>false</u>
- To declare a boolean variable, you code

  bool is_valid, was_found;
- Boolean variables quite commonly start with is_ or was_
  - Why does this make sense?
- Boolean variables are for use in <u>if</u>s

Using boolean Variables- An Example

```
string user_name, dictionary_name;
bool is_found;
…  (Missing lines that give values to user_name, dictionary_name)
if (user_name == dictionary_name)
{
    is_found = true;
}
else
{
    is_found = false;
}
…  (Several other lines)
if (is_found)
{
…  (React to the item being found)
}
```

More Powerful ifs

- The if statement is very powerful
- For example
  - The average low temperature for Fremont in September is 55.5° and the average high is 78.3°
  - We can check if a data value is in that range

```
if (temp >= 55.5   &&   temp <= 78.3)
{
    cout << “The temperature is in the usual range”
}
else
{
    cout << “The temperature is unusual”
}
```

if Logic uses: and (&&), or (||), and not (!)

- You are writing a program
  - The user will type in a number
    - You read this with cin
  - You need to verify that it's an even number between 1 and 7
  - You can code that as

```
if  (number == 2   ||   number == 4   ||   number == 6)
{
    cout << "You entered a valid number"
}
```

Operator Precedence

- Precedence refers to the order of operations
- For example, in math we do multiplication before addition
  - In programming terms, we say that multiplication has higher precedence than addition
- There is a partial precedence list in Table 5 on p. 105

Extras in the Text

- The text talks about confusing “and” and “or” on p. 107
- The text also talks about DeMorgan’s Law on p. 108
  - DeMorgan’s Law is for “not”
  - Here are two different ways to test if a number is not in the range 0-100

```
if  (!(number >= 0   &&   number <= 100))

if  (number < 0   ||   number > 100)
```

Short-circuit if Logic

- Suppose you code an if with || or &&
- As always, you need to verify that your if is correct
- You check the conditions one-by one
- Short-circuit evaluation occurs when you don't need to check all the conditions to determine the truth value of the if

Two Examples of Short-circuit Logic

```
if (homework == "done"   &&   room == "clean")
        treat = "yes";
```

- If your homework is not done, do we need to check if the room is clean?

```
if (homework == "done"   ||   room == "clean")
        treat = "yes";
```

- If your homework is done, do we need to check if the room is clean?

Input Validation

- A common problem is that when people enter data, they make mistakes
- An if statement can check that input is valid
- Most of the previous examples of complex if logic are typical input validation ifs
- Input data should always be checked to be sure it's reasonable

Input Failure

- There are times when the value entered is not correct
- One error that you have to watch for and respond to is entering the wrong type of data
- For example, you ask a person to type in his/her age
- The person is 23
- The person types: twenty-three

Responding to That Input

- C++ will get an input failure in that case
- You check using the cin.fail() function

```
int my_age;
cin >> my_age;
if (cin.fail ())
{
    cout << “Please type in a number.”;
}
```

- You usually have to clear the error status using

```
cin.clear ()
```

Other Logic Tests in C++

- C++ has two other ways to code conditional logic
- One is the switch statement
  - It is the usual way to test if a variable has one value chosen from a list
- The other is the conditional operator

These are equivalent

```
    if (temperature > 100)
    {
        fever = true;
    }
    else
    {
        fever = false;
    }
```

AND

```
  fever = Temperature > 100 ? true : false;
```

The switch Statement

- The switch is like a multi-way if
- It only allows something that evaluates to an integer to control a choice

An if Statement

```
if  (employee_code == 1)
    cout << “CEO”
else if  (employee_code == 2)
    cout << “Department Manager”
else if  (employee_code == 3)
    cout << “Programmer”
else
    cout << “Invalid code”
```

An Equivalent switch Statement

```
switch (employee_code)
{
case 1:
    cout << “CEO”
    break;
case 2:
    cout << “Department Manager”
    break;
case 3:
    cout << “Programmer”
    break;
else
    cout << “Invalid code”
}
```

Questions?

- Are there any questions?