Ch06-ConditionalExecution

August 10, 2020

1 6 Conditional Execution

1.1 Topics

- conditional executions
- comparison operators
- types of conditional statements
- switch statement
- using conditional statements in functions
- ternary conditional operator
- logical operators
- passing arguments to main and using them

1.2 6.1 Conditional execution

- so far, our programs executed top to bottom starting from **main( )**
  - statement by statement
  - functions change the execution flow from call to definition
- it's important that computer skips or executes certain block of code
  - computer needs to decide to do that to produce useful programs
- **conditional statements** let computer think or make decisions based on data
  - similar to what humans do!
  - e.g. what are the criteria/conditions that help you pick a college?
  - pick a class? go to class each day?
- conditional statements compare data values to create conditions
  - the outcome of which is true or false

1.2.1 comparison operators

- **comparison operators** are used to compare data values
  - thus, creating a condition
- comparison operators are binary operators that take two operands
- following are comparison operators that compare left hand side value with the right hand side
  - == equals to
    - already used in assert function (math symbol: =)
  - **!=** not equal to (math symbol: $\neq$)
  - > greater than
  - >= greater than or equal to (math symbol: $\geq$)
  - < less than

  - <= less than or equal to (math symbol: ≤)
- result of comparison expression (condition) is **true** or **false** boolean value
  - technically, it's **1** or **0**; **1** -> true or **0** -> false

[29]:
```
#include <iostream> // for std io
#include <cassert> // assert()
#include <string> // string
using namespace std;
```

[2]:
```
// comparison operators examples
1 == 1
```

[2]: true

[3]:
```
int x = 10;
int y = 20;
```

[4]:
```
// is x == y?
cout << (x == y);
```

0

[5]:
```
// let's print true of false using io manipulator
// is x not equal to y?
cout << boolalpha << (x != y);
```

true

[6]:
```
cout << (x > y);
```

false

[7]:
```
cout << (x < y);
```

true

[8]:
```
cout << (x >= y);
```

false

[9]:
```
cout << (x <= y);
```

true

1.3 6.2 Types of conditional statements

- there are 3 types of conditional statements:
  1. one-way selector
  2. two-way selector

3. multi-way selector

1.3.1 one-way selector

- simplest form of conditional statement
- syntax:

```
if (condition) {
    // body of if
    // block of code to execute
}
```

- the block of code inside if statement executes iff condition evalutes to true
  - skips the block, otherwise!
- the following flow-chart demonstrates the flow of if statment execution

[10]:
```
// examples
cout << "stuff before if\n";
if (true) { // true is always true; same as true == true
    cout << "body of if\n";
}
cout << "stuff after if\n";
```

```
stuff before if
body of if
stuff after if
```

[11]:
```
cout << "stuff before if\n";
if (false) { // false always evaluates to false; same as false == true
    cout << "body of if\n";
}
cout << "stuff after if\n";
```

```
stuff before if
stuff after if
```

[12]:
```
// check if a given number is positive
int num;
```

[13]:
```
cout << "enter a whole number: ";
cin >> num;
if (num > 0) {
    cout << num << " is positive\n";
}
cout << "Good bye!";
```

```
enter a whole number: 10
10 is positive
Good bye!
```

1.3.2 visualize one-way selector in pythontutor.com

1.3.3 two-way selector

- provides alternative execution
- analgoy is a true/false type question
  - you have to pick one or the other
- syntax:

```
if (condition) {
    // body of if
}
else {
    // otherwise, body of else
}
```

- if the condition is true, body of if executes
- oterwise, body of else executes
- the following flowchart demonstrates the flow of if else statement

[47]:
```
// determine if the given number is positive or negative
cout << "Enter a whole number: ";
cin >> num;
if (num > 0) {
    cout << num << " is positive\n";
}
else {
    cout << num << " is negative\n";
}
cout << "Good bye!";
```

```
Enter a whole number: -99
-99 is negative
Good bye!
```

1.3.4 visualize two-way selector in pythontutor.com

1.3.5 multi-way selector

- sometimes one may have to pick one outcome from several options
  - analagy is multiple-choice question with only one answer!
- we can achieve this by chaining a series of **if**s and **else**s
- also called chained conditionals
- syntax:

```
if (condition) {
    // first if block
}
else if(condition) {
    // 2nd if block
}
```

```
else if(condition) {
    // 3rd if block
}
...
else {
    // alternative
}
```

- check condition starting from the first **if statement**
- if the condtion is true execute the corresponding if block
  - skip the rest of the chained conditions if any
- otherwise check next condition…
- execute else alternative if not a single condition is evaluated true
- the following flowchart depicts the chained conditional execution

1.3.6 NOTE:

- since the condition is checked from top to bottom, the order of checking condition matters in some problems!

[15]:
```
// determine if a given number is 0, positive, or negative
cout << "enter a whole number: ";
cin >> num;
if (num > 0)
    // if a block has only one statment; {} can be ignored!
    cout << num << " is positive\n";
else if (num < 0)
    cout << num << " is negative\n";
else
    cout << "the entered number is 0\n";

cout << "Good bye!";
```

```
enter a whole number: 87
87 is positive
Good bye!
```

1.3.7 program that determines letter grade (A-F) given numeric grade (0-100)

- write a program that converts grade into letter grade
- letter grade criteria:

```
>= 90 ->'A'; >= 80 -> 'B'; >= 70 -> 'C'; >= 60 'D' < 60 -> 'F'
```

[16]:
```
// variable to store the value of cash in one's pocket
float grade;
```

[17]:
```
// Implementation I
// does this solution give correct answer?
// order of checking condition may matter!!
```

```
cout << "Enter a grade: ";
cin >> grade;
if (grade < 60) {
    cout << grade << "is an F!\n";
}
else if(grade >= 60) {
    cout << grade << " is a D.\n";
}
else if(grade >= 70) {
    cout << grade << "is a C.\n";
}
else if (grade >= 80) {
    cout << grade << " is a B.\n";
}
else if (grade >= 90) {
    cout << grade << " is an A!\n";
}
cout << "Good bye!";
```

```
Enter a grade: 86
86 is a D.
Good bye!
```

[18]:
```
// Implementation II
// how about this solution; does this give correct answer?
cout << "Enter a grade: ";
cin >> grade;
if (grade >= 90) {
    cout << grade << " is an A! :))\n";
    cout << "Awesome job!\n";
}
else if(grade >= 80) {
    cout << grade << " is a B. :)\n";
    cout << "Great job! So close to acing... keep working!\n";
}
else if(grade >= 70) {
    cout << grade << " is a C. :|\n";
    cout << "Good job! work harder to get a B or an A\n";
}
else if(grade >= 60) {
    cout << grade << " is a D. :(\n";
    cout << "Sorry, D isn't good enought to move on to CS2\n. Work very hard!!";
}
else {
    cout << grade << " is an F. :((\n";
    cout << "Sorry, that's a fail. Work really really hard to pass!!\n";
}
```

```
cout << "Good bye!\n";
```

```
Enter a grade: 75
75 is a C. :|
Good job! work harder to get a B or an A
Good bye!
```

[19]:
```
// Implementation III - using function
char find_letter_grade(float grade) {
    if (grade >= 90)
        return 'A';
    else if(grade >= 80)
        return 'B';
    else if(grade >= 70)
        return 'C';
    else if(grade >= 60)
        return 'D';
    else
        return 'F';
}
```

[20]:
```
// manually test function
cout << "Enter a grade: ";
cin >> grade;
cout << grade << " is equivalent to " << find_letter_grade(grade);
```

```
Enter a grade: 100
100 is equivalent to A
```

[21]:
```
// write at least three automated test cases
void test_find_letter_grade() {
    assert(find_letter_grade(100) == 'A');
    assert(find_letter_grade(40) == 'F');
    assert(find_letter_grade(89) == 'B');
    // TODO: test for every possible outcome
    cout << "all test casses passed!" << endl;
}
```

[22]:
```
test_find_letter_grade();
```

```
all test casses passed!
```

1.3.8 visualize multi-way selector in pythontutor.com

1.4 6.3 Nested conditionals

- one or more type of conditional statements can be nested inside conditional statements
- syntax:

```
if (condition) {
    // do something
    if (condition) {
        // do something..
    }

    if (condition) {
        // do something
    }
    else {
        // do something else
    }

}
else {
    // do something else...
    if (condition) {
        // do something
    }
}
```

[23]:
```
// a program that determines if a given number is 0, even or odd and positive␣
 ↪or negative
// the order of condition doesn't matter in this example
cout << "enter a whole number: ";
cin >> num;
if (num > 0) {
    cout << num << " is positive ";
    // check if the number is even or odd
    if (num %2 == 0)
        cout << "and even\n";
    else
        cout << "and odd\n";
}
else if (num < 0) {
    cout << num << " is negative ";
    // check if the number is even or odd
    if (num %2 == 0)
        cout << "and even\n";
    else
        cout << "and odd\n";
}
else
    cout << "the entered number is 0\n";

cout << "Good bye!";
```

```
enter a whole number: 3
```

```
3 is positive and odd
Good bye!
```

[24]:
```
// TODO: Convert the above program as a function
```

1.4.1 visualize nested conditional execution in pythontutor.com

1.5 6.4 Conditional operator

- C++ provies a ternary conditional operator
- takes 3 operands
- syntax:

```
(Condition) ? Exp2 : Exp3;
```

- the value of (Condition) is evaluated
- if the Condition is true, Exp2 is used as the result
- otherwise Exp3 is uesed as the result or the operator
- simply a shortcut for:

```
if (Condition) {
    var = Exp2;
}
else {
    var = Exp3;
}
```

[31]:
```
// application of conditional operator
// write a program that determines if a given number is odd or even

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

// declare num if need be
//int num;
```

[30]:
```
cout << "Enter a whole number: ";
cin >> num;
cout << num << " is " << ((num%2 == 0) ? "even" : "odd");
```

```
Enter a whole number: 10
10 is even
```

1.6 6.5 Logical operators

- often times programs need to evaluate complex logics involving two or more logical expressions
- C++ provides three logical operators to evaluate complex boolean expressions
    1. && (to ampersands; read as **and**)

2. | | (two pipes; read as **or**)
3. **!** (bang or exclamation; read as **not**)

- && and || are binary operators
- ! is an unary operator
- let's say if **a** and **b** are logical expression resulting **true (T)** or **false (F)**
  - the following truth table provides the final outcome of these logical operators

1.6.1 Truth table for && (and)

a	b	a && b
T	T	T
T	F	F
F	T	F
F	F	F

1.6.2 Truth table for || (or)

a	b	a || b
T	T	T
T	F	T
F	T	T
F	F	F

1.6.3 Truth table for ! (not)

a	! a
T	F
F	T

1.6.4 Order of evalution

- if all three operators are found in the same expression:
  - ! is evaluated first, && second and finally ||
- complete C++ operator precedence order can be found here: https://en.cppreference.com/w/cpp/language/operator_precedence

[33]:
```
// && examples
// determine if a number is even and positve
cout << "enter a whole number: ";
cin >> num;
if (num > 0 && num%2 == 0)
    cout << "number is even and positve\n";
```

```
else
    cout << "I don't know much about " << num << " except that it's an␣
 ↪integer\n";
```

```
enter a whole number: 100
number is even and positve
```

[37]:
```
// || or example
// write a program that determines if somone can retire.
// if a person owns a Ferrari or has 1 Million dollors in savings then the␣
 ↪person can retire
string has_ferrari;
long savings;
```

[38]:
```
cout << "Do you own a Ferarrai? Enter [y|yes]: ";
cin >> has_ferrari;
cout << "How much in savings do you have in dollars? ";
cin >> savings;
if (has_ferrari == "yes" or has_ferrari == "y" or savings >= 1000000)
    cout << "Congratulations, you can retire now!\n";
else
    cout << "Sorry, no cigar! Keep working...\n";
```

```
Do you own a Ferarrai? Enter [y|yes]: yes
How much in savings do you have in dollars? 10
Congratulations, you can retire now!
```

[40]:
```
// ! example
// redo retirement calculator
cout << "Do you own a Ferarrai? Enter [y|yes]: ";
cin >> has_ferrari;
cout << "How much in savings do you have in dollars? ";
cin >> savings;
if (!(has_ferrari == "yes" or has_ferrari == "y" or savings >= 1000000))
    cout << "Sorry, no cigar! Keep working...\n";
else
    cout << "Congratulations, you can retire now!\n";
```

```
Do you own a Ferarrai? Enter [y|yes]: n
How much in savings do you have in dollars? 10
Sorry, no cigar! Keep working…
```

1.7 6.6 Passing arguments to main

- main( ) can also take arguments
- since main is never called, arguments are provided when the program is ran from a terminal
- the program doesn't have to interactively prompt user to enter required data
- syntax:

```
int main(int argc, char* argv[]) {
    // argc is total no. of arguments provided to the program
    // automatically calcuated by the system based on the no. of arguments
    // argc is atleast 1
    // argv is an array of char* (c_string; similar in concept to C++ string)
    // contains name of the program and all the user provided arguments

    // body of main
    return 0;
}
```

- pass space separated arguments to main or program
- use double quotes for arguments with spaces
- all the arguments are treated as c-string

```
$ programName.exe arg1 arg2 arg3 "multiple word arguments" ...
```

1.7.1 demo programs

1. first see demo_programs/Ch06/main_arg.cpp
2. more useful application: demo_programs/Ch06/main_arg1.cpp
3. Kattis Hello World problem with test case: https://github.com/rambasnet/KattisDemos/blob/master/hello/

1.8 6.7 Switch statement

- switch statment is very similar to chained conditional or multi-way selector
- allows a variable to be tested for equality against a list of values
- each value is called a case
- syntax:

```
switch(integral-expression) {
    case constant-expression:
        statement(s);
        break; // optional
    case constant-expression:
        statements(s);
        break; // optional
    // more case statements
    default: // Optional
        statements(s);
}
```

- switch only works on integral type value
- when break statement is reached, switch terminates
- if no break statement exists, the statements following that case will execute until a break statement is reached
- the following figure demonstrates the flow of execution in switch statement

1.8.1 Menu-driven CLI interface

- command-line interface (CLI), though not as intuitive as Graphical User Interface (GUI), is still used commonly
- airline reservation systems, check-in and printing boarding passes, point-of-sale (POS) terminals at big companies such as Lowe's and Home Depot use CLI
- a lot of text-based games used CLI
- a good application of switch statement is in developing menu-driven CLI

1.8.2 write a menu-driven C++ program that calculates various statistics of any 2 numbers

[1]:
```
#include <iostream>
#include <string>
#include <cassert>
#include <cmath>
#include <iomanip>
#include <sstream>

using namespace std;
```

[2]:
```
template<class T>
T add(T val1, T val2) {
    return val1 + val2;
}
```

[3]:
```
template<class T>
T subtract(T val1, T val2) {
    return val1 - val2;
}
```

[4]:
```
template<class T>
T larger(T val1, T val2) {
    return val1 >= val2?val1:val2;
}
```

[5]:
```
template<class T>
double average(T val1, T val2) {
    return add(val1, val2)/2.0;
}
```

[6]:
```
int getMenuOption() {
    // A Smiple CLI-based calculator
    int option;
    cout << "Enter one of the following menu options: [1-6]\n"
        "1 -> Add\n"
```

```
        << "2 -> Subtract\n"
        << "3 -> Larger\n"
        << "4 -> Average\n"
        << "5 -> Multiply\n"
        << "6 -> Quit\n";
    cin >> option;
    return option;
}
```

[7]:
```
void program() {
    float n1, n2;
    int option;
    option = getMenuOption();
    if (option == 6) {
        cout << "Good bye...\n";
        return;
    }
    cout << "Enter two numbers separated by space: ";
    cin >> n1 >> n2;
    switch(option) {
        case 1:
            cout << n1 << " + " << n2 << " = " << add<float>(n1, n2) << endl;
            break; // terminate switch
        case 2:
            cout << n1 << " - " << n2 << " = " << subtract<float>(n1, n2) <<␣
 ↪endl;
            break;
        case 3:
            cout << "larger between: " << n1 << " and " << n2 << " is " <<␣
 ↪larger<float>(n1, n2) << endl;
            break;
        case 4:
            cout << "average of " << n1 << " and " << n2 << " = " <<␣
 ↪average<float>(n1, n2) << endl;
            break;
        default:
            cout << n1 << " x " << n2 << " = " << n1*n2 << endl;
            break;
    }
}
```

[10]:
```
// TODO: run this many times...
program();
```

```
Enter one of the following menu options: [1-6]
1 -> Add
2 -> Subtract
```

```
3 -> Larger
4 -> Average
5 -> Multiply
6 -> Quit
5
Enter two numbers separated by space: 100 99
100 x 99 = 9900
```

1.8.3 Note: a loop would work better for menu-driven program

- which is covered in next chapter

1.9 6.8 Exercises

1. Write a program that helps someone decide where to go eat lunch depending on amount of money one has in their pocket.
2. Improve exercise 1 by using function(s) and writing at least 3 test cases for each function.
3. Write a program that determines whether someone is eligible to vote in the US federal election.
   - see sample solution here demo_programs/Ch06/voting_eligibility.cpp
4. Improve exercise 3 by using function(s) and writing at least 3 test cases for each function.
   - see sample solution here demo_programs/Ch06/voting_eligibility_v2.cpp
5. Write a function day_name that converts an integer number 0 to 6 into the name of a day. Assume day 0 is "Sunday". Return "Invalid Day" if the argument to the function is not valid.

[ ]:
```
// code stub for Exercise 5
string day_name(int day) {
    // FIXME - complete the rest
}
```

[ ]:
```
// Here are some tests that should pass for day_name function defined above
void test_day_name() {
    assert(day_name(3) == "Wednesday");
    assert(day_name(6) == "Saturday");
    assert(day_name(42) == "Invalid Day");
    cout << "all test cases passed for day_name()\n";
}
```

6. Improve exercise 5 as a complete program with algorithm stepts, main(), etc.
7. Write a function that helps answer questions like "Today is Wednesday. I leave on holiday in 19 days time. What day will that be?" So, the function must take a day name and a delta argument (the number of days to add) and should return the resulting day name.

[ ]:
```
// Exercise 6 hints
string day_add(string dayName, int delta) {
    // FIXME
```

```
}
```

[48]:
```
// Exercise 6 test function
// here are some tests that should pass
void test_day_add() {
    assert(day_add("Monday", 4) ==  "Friday");
    assert(day_add("Tuesday", 0) == "Tuesday");
    assert(day_add("Tuesday", 14) == "Tuesday");
    assert(day_add("Sunday", 100) == "Tuesday");
    assert(day_add("Sunday", -1) == "Saturday");
    assert(day_add("Sunday", -7) == "Sunday");
    assert(day_add("Tuesday", -100) == "Sunday");
    cout << "all test cases passed for day_add()";
}
```

8. Improve Exercise 7 as a complete program with algorithm steps, main(), etc.
9. Write a C++ program including algorithm steps that calculates area and perimeter of a triangle given three sides.
    - must define and use separate functions to calculate area and perimeter
    - write at least 3 test cases for each function
    - Hint: use Heron's formula to find area with three sides.
    - a partial solution is provided here demo_programs/Ch04/triangle.cpp
    - **TODO: improve the program; define and use function to determine if 3 sides form a triangle**
10. Write a C++ program including algorithm steps that calculates Body Mass Index (BMI) of a person.
    - must use as many functions as possible
    - write at least 3 test cases for each function
    - more info on BMI - https://www.nhlbi.nih.gov/health/educational/lose_wt/BMI/bmicalc.htm
    - Formula here.
    - a sample solution is provided at demo_programs/Ch06/BMI_v3.cpp
    - **improved version that interprets the BMI result**

1.10 6.9 Kattis Problems

- there are not many Kattis problems that utilizes only the concepts covered so far
- almost all of them utilize the concept covered so far but that's not enought to solve those problems

1.11 6.10 Summary

- we learned about another fundamental concepts: conditional execution
- learned with examples 3 different types of conditional statements
- learned how to use conditional statements in functions
- learned about ternary conditional operator (condition) ? exp1 : exp2
    - a short cut for alternative execution

- learned about comparision and logical operators; order of precedence
- learned passing and using arguments to main( )
- finally, exercise and sample solutions

[ ]: