Laboratory 4A

CS-102

Spring 2022

Laboratory 4A – Part 1

Note: Before beginning this lab, read it through entirely. You may find the material in the appendices to be of significant help to you.

- Your task, for Laboratory 4A is to first type in Program4-18, and get it running.
- When you have it working OK, call the instructor over so you can be credited for not only making it operational, but for <u>using proper Gaddis formatting</u>.
- Call your program: *YourName*-Lab04A_1.cpp .

Laboratory 4A – Part 2

- Now you will make some changes to Program4-18.
- You will change the **if else/if** statement to the **switch** statement, and get it running that way.
- Tip: some things to look out for:
    - In a switch statement structure you need a **break** statement at the end of each **case** block, whereas you didn't need this in the **if else/if** structure.
    - Be careful of the use of braces and colons when using the two structures. They are different.
- <u>Make certain that you have used the proper Gaddis formatting.</u>
    - Make sure that your indentations are correct.
    - Make sure your opening and closing braces are aligned in the same column.
    - Make certain that your lines, starting with the word, "**case**", are aligned in the same column.
- When you get Program4-18 rewritten and working using the switch structure, Call your program: *YourName*-Lab04A_2.cpp
- If you are doing Lab04A synchronously, ask the instructor to check out your work so that you can be given proper credit.
- If you are doing Lab04A asynchronously, submit your program to Canvas.

Laboratory 4A – Parts 3&4

- We are going to write a program that inputs a letter grade and displays a message as follows:

Letter grade	Message
A	Excellent
B	Above average
C	Average
D	Below average
F	Below average

- Write this program first using the **if/else if** structure.
  - Call your program: *YourName*-Lab04A_3.cpp .
  - <u>Pay close attention to your formatting.</u>
  - <u>Be sure to allow for either lower case or upper case grades to be input.</u>
- Then write this program using the **switch/case** structure.
  - Call your program: *YourName*-Lab04A_4.cpp .
  - <u>Pay particular attention to your formatting and make input case-independent.</u>
  - See if you can make use of the “fall through” property to combine cases ‘D’ & ‘F’.

Laboratory 4A – Parts 3&4

- When you get *YourName*-Lab04A_4.cpp working:
  - If you are doing Lab04A synchronously, ask the instructor to check out your work so that you can be given proper credit.
  - If you are doing Lab04A asynchronously, submit your programs to Canvas.

Laboratory 04A Part 5

- In the appendix to this Laboratory, you will see the use of: if (cin.fail()) to test to be certain that we are only inputting numbers into our program, not alpha characters.
- The only problem with the if (cin.fail()) statement is that execution is terminated if a person accidentally strikes an alpha key instead of a numeric key.
- This problem can be solved by using: while (cin.fail()), followed by code within the braces that invites the user to re-enter their response using a number instead of an alpha character.
- In this way program termination can be avoided.

Laboratory 04A Part 5

- Rewrite the program called elevator.cpp, substituting while (cin.fail()) for if (cin.fail()) and then replacing the code that is within the braces below, allowing the user to re-type in a numeric quantity that is acceptable to the program.
- Caution: The first two lines within the braces that follow your line containing while (cin.fail()) must contain the following two statements:

```
cin.clear();
cin.ignore();
```

  - These two lines will reset the failbit and will allow you to move beyond this failure.
  - The remaining lines should give your error message followed by a retry of typing in a valid floor number.
- Call your program: *YourName*-Lab04A_5.cpp
- When you get *YourName*-Lab04A_5.cpp working:
  - If you are doing Lab04A synchronously, ask the instructor to check out your work so that you can be given proper credit.
  - If you are doing Lab04A asynchronously, submit your programs to Canvas.

Appendices

Menus

- Menu-driven program: program execution controlled by user selecting from a list of actions
- Menu: list of choices on the screen
- Menus can be implemented using `if/else if` statements

Menu-Driven Program Organization

- Display list of numbered or lettered choices for actions
- Prompt user to make selection
- Test user selection in `expression`
    - if a match, then execute code for action
    - if not, then go on to next `expression`

```
// Program 4-18
// This program displays a menu and asks the user to make a selection.
// An if/else if statement determines which item the user has chosen.
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
  int choice;      // To hold a menu choice
  int months;      // To hold the number of months
  double charges;  // To hold the monthly charges
  // Constants for membership rates
  const double          ADULT = 40.0,
                        SENIOR = 30.0,
                        CHILD = 20.0;
  // Constants for menu choices
  const int      ADULT_CHOICE = 1,
                 CHILD_CHOICE = 2,
                  SENIOR_CHOICE = 3,
                  QUIT_CHOICE = 4;
```

```
// Display the menu and get a choice.
   cout << "\t\tHealth Club Membership Menu\n\n";
   cout << "1. Standard Adult Membership\n";
   cout << "2. Child Membership\n";
   cout << "3. Senior Citizen Membership\n";
   cout << "4. Quit the Program\n\n";
   cout << "Enter your choice: ";
   cin >> choice;

   // Set the numeric ouput formatting.
   cout << fixed << showpoint << setprecision(2);
```

```
if (choice == ADULT_CHOICE)
  {
    cout << "For how many months? ";
    cin >> months;
    charges = months * ADULT;
    cout << "The total charges are $" << charges << endl;
  }
  else if (choice == CHILD_CHOICE)
  {
    cout << "For how many months? ";
    cin >> months;
    charges = months * CHILD;
    cout << "The total charges are $" << charges << endl;
  }
  else if (choice == SENIOR_CHOICE)
  {
    cout << "For how many months? ";
    cin >> months;
    charges = months * SENIOR;
    cout << "The total charges are $" << charges << endl;
  }
```

```
else if (choice == QUIT_CHOICE)
   {
       cout << "Program ending.\n";
   }
   else
   {
      cout << "The valid choices are 1 through 4. Run the\n";
      cout << "program again and select one of those.\n";
   }
   return 0;
}
```

The **`switch`** Statement

- Used to select among statements from several alternatives
- In some cases, can be used instead of `if/else if` statements

switch Statement Format

```
switch (expression) //integer
{
 case exp1: statement1;
 case exp2: statement2;
 ...
 case expn: statementn;
 default:   statementn+1;
}
```

`break` Statement

- Used to exit a `switch` statement
- If it is left out, the program "falls through" the remaining statements in the `switch` statement

Program 4-23 Example of **switch** Statement

```
// The switch statement in this program tells the user something
// he or she already knows: what they just entered!
#include <iostream>
using namespace std;
int main()
{
    char choice;
    cout << "Enter A, B, or C: ";
    cin >> choice;
    switch (choice)
    {
        case 'A':
            cout << "You entered A.\n";
            break;
        case 'B':
            cout << "You entered B.\n";
            break;
        case 'C':
            cout << "You entered C.\n";
            break;
        default:
            cout << "You did not enter A, B, or C!\n";
    }
    return 0;
}
```

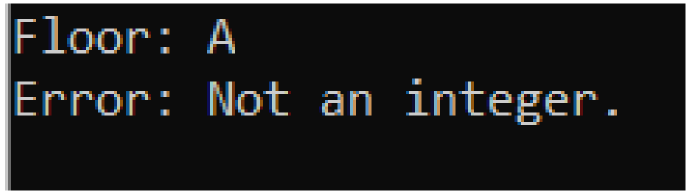
Avoiding Runtime Errors Using Cin

- As we know, cin has the problem that it can’t read in a space character.
- One solution is to limit cin to only processing numeric quantities.
- A means of doing this is to use the cin.fail() test;
- The instruction: if (cin.fail()) will return a true if what is typed in is numeric, otherwise it will return a false.
- This is most useful if you want to be sure the user is only typing in numbers.
- If you insert the following code into a program, you can catch a user trying to type in an alpha character when only a numeric character is permitted. This will stop program execution thus preventing a runtime error.

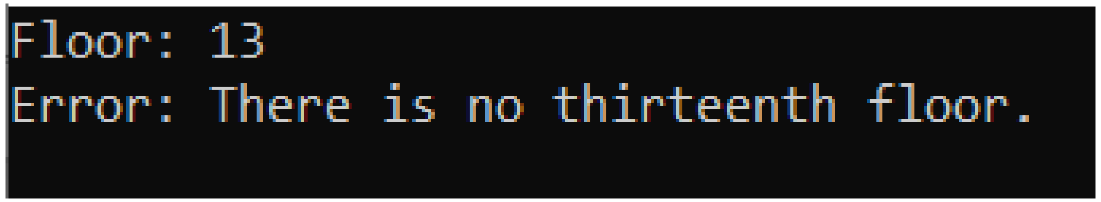
```
if (cin.fail())
{
        cout << "Error: Not an integer." << endl;
        return 1;
}
```

```
#include <iostream>
using namespace std;
int main()
{
  int floor;
  cout << "Floor: ";
  cin >> floor;
  // The following statements check various input errors
  if (cin.fail())
  {
    cout << "Error: Not an integer." << endl;
    return 1;
  }
  if (floor == 13)
  {
    cout << "Error: There is no thirteenth floor." << endl;
    return 1;
  }
  if (floor <= 0 || floor > 20)
  {
    cout << "Error: The floor must be between 1 and 20." << endl;
    return 1;
  }
```

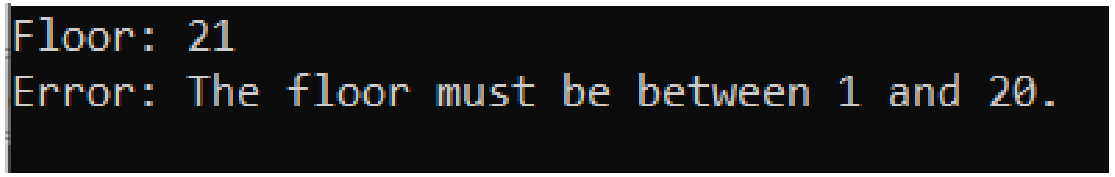
Elevator.cpp Checks for input errors. Page 1 of 2



```
Floor: A
Error: Not an integer.
```



```
Floor: 13
Error: There is no thirteenth floor.
```



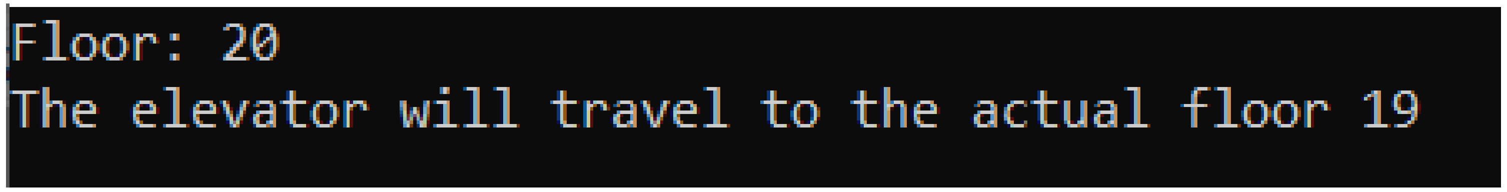
```
Floor: 21
Error: The floor must be between 1 and 20.
```

Elevator.cpp Checks for input errors. Page 2 of 2

```
// Now we know that the input is valid
  int actual_floor;
  if (floor > 13)
  {
    actual_floor = floor - 1;
  }
  else
  {
    actual_floor = floor;
  }

  cout << "The elevator will travel to the actual floor "
    << actual_floor << endl;

  return 0;
}
```



```
Floor: 20
The elevator will travel to the actual floor 19
```