Software and errors

Michael Nowak

Texas A&M University

Acknowledgement: Some lecture slides based on those created by Bjarne Stroustrup for use with his textbook

Notes

Overview

Notes

Overview

Notes

Software

- Software are the programs that run on the hardware
- Like hardware, can be seen as having multiple components:
  - The *BIOS (basic input/oputput system)* is the base layer that provides computer initial instructions for what to do when powered on
  - *Operating system* is responsible for controlling the operations of the machine, how the user interacts with it, reading/writing files to disk, and loading and starting other programs
  - *Application and utility programs* are those that the user runs, such as your email client or web browser

Notes

Overview

Notes

Nature of programming

- Every piece of software is written by a programmer, but
  - what is programming, and
  - how do we do it?
- At the fundamental level, during each cycle, the computer loads an instruction and executes it

Notes

Overview

Notes

Machine language

- Each instruction is encoded as a binary sequence of numbers; the language of these instructions is known as *machine language*
- For instance, using the MIPs machine language, we could write the equation `wage = rate * hours` as:

```
100011 00000 00010 0000000000000000   # Load rate, register 2
100011 00001 00011 0000000000000000   # Load hours, register 3
000000 00010 00011 00100 00000 011000 # Multiply registers 2 and 3;
                                        store the result in register 4
101011 00100 00101 0000000000000000   # Store value of register 4
```

Notes

Overview

Notes

Assembly language

- Assembly language has an assembly instruction for each machine language instruction
- Unlike machine language, assembly language is entered as mnemonics (i.e., words) that describe what they do
- For instance, we could write the equation `wage = rate * hours` as:

```
lw   $s0, $s2, 0
lw   $s1, $s3, 0
mult $s2, $s3, $s4
sw   $s4, $s5, 0
```

- In order for the assembly language to be understood by the computer, we use an *assembler* to translate from assembly language to machine language

Notes

Overview

Notes

Higher-level languages

- It is hard for a programmer to express ideas in machine language and assembly language
- Higher-level languages use more complete mnemonics and allow more complex organization of ideas
- In C++, provided that `wage` had been *declared*, and `rate` and `hours` had been *defined*, we could simply write the following *statement* in our program:

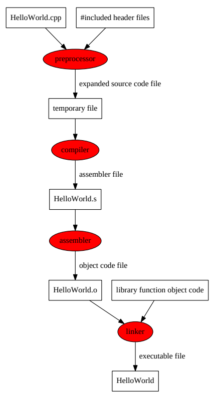
```
wage = rate * hours;
```

Notes

Overview

Notes

C++ Compilation Processes



Notes

Overview

Notes

Errors

- When we write programs, errors are natural and unavoidable; the question is, how do we deal with them?
  - Organize software to minimize errors
  - Eliminate most of the errors we made anyway
    - Debugging
    - Testing

"My guess is that avoiding, finding, and correcting errors is 95% or more of the effort for serious software development."
– Bjarne Stroustrup

Notes

Overview

Notes

Sources of errors

- Poor specification
  - "What s this suppose to do?"
- Incomplete programs
  - "but I ll get around to it... tomorrow..."
- Unexpected arguments to functions
  - "but `sqrt()` isn t suppose to be called with `-1` as its argument"
- Unexpected input
  - "but the user was suppose to input an integer"
- Code that simply doesn t do what it was supposed to do
  - "so fix it..."

Notes

Overview

Notes

Your program

- Should produce the desired results for all legal inputs
- Should give reasonable error messages for all illegal inputs
- Need not worry about misbehaving hardware
- Need not worry about misbehaving system software
- Is allowed to terminate after finding an error

Notes

Overview

Notes

Kinds of errors

Compile-time errors Errors found by the compiler
- Syntax errors
- Type errors

Link-time errors Errors found by the linker when it is trying to combine object files into an executable program

Run-time errors Errors found by checks made during a running program; that is, errors detected by
- the computer (hardware and/or the operating system)
- by a library (e.g., the standard library)
- by user code

Logic errors Errors found by the programmer looking for the causes of erroneous results

Notes

Overview

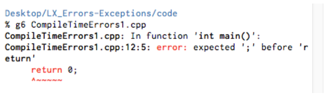
Notes

Overview

Notes

Compile-time errors : Syntax errors

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

int main ( ) {
    string first_name = "Michael";
    string last_name = "Nowak";
    string full_name = first_name + '␣' + last_name;
    cout << full_name << endl

    return 0;
}
```



```
Desktop/LX_Errors-Exceptions/code
% g6 CompileTimeErrors1.cpp
CompileTimeErrors1.cpp: In function 'int main()':
CompileTimeErrors1.cpp:12:5: error: expected ';' before 'r
eturn'
     return 0;
     ^~~~~~
```

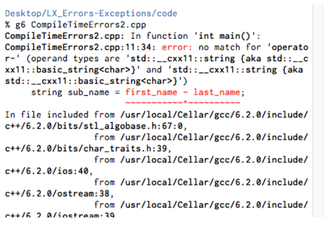
Notes

Overview

Notes

Compile-time errors : Type errors

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

int main ( ) {
    string first_name = "Michael
    string last_name = "Nowak";


    string sub_name = first_name − last_name;
    cout << sub_name;

    return 0;
}
```



```
Desktop/LX_Errors-Exceptions/code
% g6 CompileTimeErrors2.cpp
CompileTimeErrors2.cpp: In function 'int main()':
CompileTimeErrors2.cpp:11:34: error: no match for 'operato
r-' (operand types are 'std::__cxx11::string {aka std::__c
xx11::basic_string<char>}' and 'std::__cxx11::string {aka
std::__cxx11::basic_string<char>}')
     string sub_name = first_name - last_name;
                       ~~~~~~~~~~~^~~~~~~~~~~
In file included from /usr/local/Cellar/gcc/6.2.0/include/
c++/6.2.0/bits/stl_algobase.h:67:0,
                 from /usr/local/Cellar/gcc/6.2.0/include/
c++/6.2.0/bits/char_traits.h:39,
                 from /usr/local/Cellar/gcc/6.2.0/include/
c++/6.2.0/ios:40,
                 from /usr/local/Cellar/gcc/6.2.0/include/
c++/6.2.0/ostream:38,
                 from /usr/local/Cellar/gcc/6.2.0/include/
c++/6.2.0/iostream:39
```

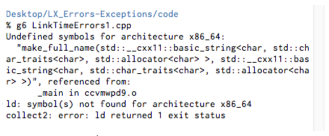
Notes

Overview

Notes

Link-time errors

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

/*
    declaration, for an undefine
*/
string make_full_name (string f, string l);


int main ( ) {
    string first_name = "Michael";
    string last_name = "Nowak";
    string full_name = make_full_name(first_name, last_name);

    return 0;
}
```



```
Desktop/LX_Errors-Exceptions/code
% g6 LinkTimeErrors1.cpp
Undefined symbols for architecture x86_64:
  "make_full_name(std::__cxx11::basic_string<char, std::ch
ar_traits<char>, std::allocator<char> >, std::__cxx11::bas
ic_string<char, std::char_traits<char>, std::allocator<cha
r> >)", referenced from:
      _main in ccvmwpd9.o
ld: symbol(s) not found for architecture x86_64
collect2: error: ld returned 1 exit status
```

Notes

Overview

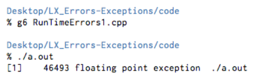
Notes

Overview

Notes

Run-time errors : detected by the computer

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

int main ( ) {

    int x = -1;
    int y = 0;
    /*
        divide by zero
    */
    int z = x / y;
    cout << z;

    return 0;
}
```



```
Desktop/LX_Errors-Exceptions/code
% g6 RunTimeErrors1.cpp

Desktop/LX_Errors-Exceptions/code
% ./a.out
[1]    46493 floating point exception  ./a.out
```

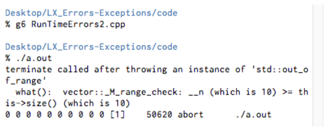
Notes

Overview

Notes

Run-time errors : detected by a library

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

 int main ( ) {

     vector<int> v(10);
     /*
         when we are at v.size(), we are out of
         v's range of elements
     */
     for (int i = 0 ; i <= v.size() ; ++i)
         cout << v.at(i) << '␣';

     return 0;
 }
```



```
Desktop/LX_Errors-Exceptions/code
% g6 RunTimeErrors2.cpp

Desktop/LX_Errors-Exceptions/code
% ./a.out
terminate called after throwing an instance of 'std::out_o
f_range'
  what():  vector::_M_range_check: __n (which is 10) >= th
is->size() (which is 10)
0 0 0 0 0 0 0 0 0 0 [1]    50620 abort      ./a.out
```

Notes

Overview

Notes

Run-time errors : detected by user-code

- We can find errors through various checks made during a running program...

Notes

Overview

Notes

Local run-time errors

- Easy to do for local run-time errors

  -
    ```
    int i;
    std::cin >> i;
    if (i < 0)
        return 1;
    ```

Notes

Overview

Notes

Non-local run-time errors

- How can we handle non-local errors during run-time?

```
// necessary #includes...

int area (int length, int width) { return length * width; }
int framed_area (int x, int y) { return area(x-2, y-2); }

int main ( ) {
    int x = -1;
    int y = 2;
    int z = 4;
    // ...
    int area1 = area(x, y);
    int area2 = framed_area(1, z);
    int area3 = framed_area(y, z);
    double ratio = double(area1)/area3;
    return 0;
}
```

- Need some means of error reporting... will discuss this shortly

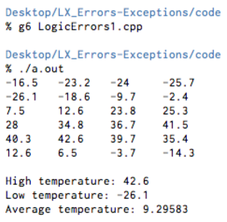
Notes

Overview

Notes

Logic errors

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

int main ( ) {

    vector<double> temps {-16.5, -23.2, -24.0, -25.7, -26.1, -18.6, -9.7, -2.4,
        7.5, 12.6, 23.8, 25.3, 28.0, 34.8, 36.7, 41.5, 40.3, 42.6, 39.7, 35.4,
        12.6, 6.5, -3.7, -14.3};

    double sum = 0;
    double high_temp = 0;
    double low_temp = 0;

    for (double t : temps) {
        if (t > high_temp) high_temp = t;
        if (t < low_temp) low_temp = t;
        sum += t;
    }

    double avg_temp = sum/temps.size();
    for (int i = 1 ; i <= temps.size() ; ++ i) {
        cout << temps.at(i-1) << '\t';
        if (i % 4 == 0) cout << endl;
    }
    cout << endl;
    cout << "High␣temperature:␣" << high_temp << endl;
    cout << "Low␣temperature:␣" << low_temp << endl;
    cout << "Average␣temperature:␣" << avg_temp << endl;

}
```

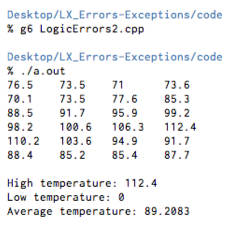


```
Desktop/LX_Errors-Exceptions/code
% g6 LogicErrors1.cpp

Desktop/LX_Errors-Exceptions/code
% ./a.out
-16.5   -23.2   -24     -25.7
-26.1   -18.6   -9.7    -2.4
7.5     12.6    23.8    25.3
28      34.8    36.7    41.5
40.3    42.6    39.7    35.4
12.6    6.5     -3.7    -14.3

High temperature: 42.6
Low temperature: -26.1
Average temperature: 9.29583
```

Notes

Logic errors

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

int main ( ) {

    vector<double> temps {76.5, 73.5, 71.0, 73.6, 70.1, 73.5, 77.6, 85.3, 88.5,
        91.7, 95.9, 99.2, 98.2, 100.6, 106.3, 112.4, 110.2, 103.6, 94.9, 91.7,
        88.4, 85.2, 85.4, 87.7};

    double sum = 0;
    double high_temp = 0;
    double low_temp = 0;

    for (double t : temps) {
        if (t > high_temp) high_temp = t;
        if (t < low_temp) low_temp = t;
        sum += t;
    }

    double avg_temp = sum/temps.size();
    for (int i = 1 ; i <= temps.size() ; ++ i) {
        cout << temps.at(i-1) << '\t';
        if (i % 4 == 0) cout << endl;
    }
    cout << endl;
    cout << "High␣temperature:␣" << high_temp << endl;
    cout << "Low␣temperature:␣" << low_temp << endl;
    cout << "Average␣temperature:␣" << avg_temp << endl;

}
```

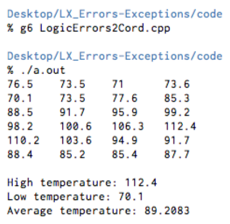


```
Desktop/LX_Errors-Exceptions/code
% g6 LogicErrors2.cpp

Desktop/LX_Errors-Exceptions/code
% ./a.out
76.5    73.5    71      73.6
70.1    73.5    77.6    85.3
88.5    91.7    95.9    99.2
98.2    100.6   106.3   112.4
110.2   103.6   94.9    91.7
88.4    85.2    85.4    87.7

High temperature: 112.4
Low temperature: 0
Average temperature: 89.2083
```

Notes

Logic errors

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

int main ( ) {

    vector<double> temps {76.5, 73.5, 71.0, 73.6, 70.1, 73.5, 77.6, 85.3, 88.5,
        91.7, 95.9, 99.2, 98.2, 100.6, 106.3, 112.4, 110.2, 103.6, 94.9, 91.7,
        88.4, 85.2, 85.4, 87.7};

    double sum = 0;
    double high_temp = temps[0];
    double low_temp = temps[0];

    for (double t : temps) {
        if (t > high_temp) high_temp = t;
        if (t < low_temp) low_temp = t;
        sum += t;
    }

    double avg_temp = sum/temps.size();
    for (int i = 1 ; i <= temps.size() ; ++ i) {
        cout << temps.at(i-1) << '\t';
        if (i % 4 == 0) cout << endl;
    }
    cout << endl;
    cout << "High␣temperature:␣" << high_temp << endl;
    cout << "Low␣temperature:␣" << low_temp << endl;
    cout << "Average␣temperature:␣" << avg_temp << endl;

}
```



```
Desktop/LX_Errors-Exceptions/code
% g6 LogicErrors2Cord.cpp

Desktop/LX_Errors-Exceptions/code
% ./a.out
76.5    73.5    71      73.6
70.1    73.5    77.6    85.3
88.5    91.7    95.9    99.2
98.2    100.6   106.3   112.4
110.2   103.6   94.9    91.7
88.4    85.2    85.4    87.7

High temperature: 112.4
Low temperature: 70.1
Average temperature: 89.2083
```

Notes

Overview

Notes

References

- Lippman, B., Lajoie, Josee, & Moo, B. E. (2016). *C++ primer* (5th ed.). Addison-Wesley.
- Stroustrup, B. (2014). *Programming: principles and practice using C++* (2nd ed.). Addison-Wesley.

Notes

Notes

Notes