MH1402 Algorithms & Computing II

Lecture 12 Revision

Wu Hongjun

Overview

- **Revision**
- **Common errors in past year exams (including our midterm exam)**
- **Information on Final Exam**

What have we learned in this course?

- **C++**
  - **The focus of this course**
  - **We learned the most essential features of C++**

- **Elementary algorithms**
  - **Search, sorting**
  - **Mainly for practicing C++**

High-Level Programming Languages

- **Two main types of high-level programming languages**
  - **Interpreted languages**
    - **The code get read then executed directly by the interpreter**
    - **Example: Matlab, Perl, Python, …**
  - **Compiled languages**
    - **The code should be compiled and linked into an executable program; then the executable program can run without involving the compiler**
    - **Example: C, C++, Fortran, ….**

High-Level Programming Languages

- **What are common/essential for high-level programming languages?**
  - **Variables**
  - **Arithmetic Operators**
  - **Comparison & Logical Operators**
  - **Control Statements**
    - **Selection statement: if, if-else,**
    - **Loop statement: for, while, do…while,**
  - **Arrays**
    - **The most basic and important data structure**
  - **Functions**
    - **Passing by value, passing by reference**
  - **Input/Output**
    - **Standard input/output (keyboard and screen)**
    - **File input/output**

C++

- **C++ is a high-level programming language**
  - **It has all the essential components listed in the previous slide**
- **C++ provides the following additional features**
  - **Pointers**
  - **Classes (the main difference between C and C++)**

C++: Data type and Variables

- **Integers**
  - **int, unsigned int (at least 32-bit for today's compilers)**
  - **long long int, unsigned long long int (at least 64-bit)**
- **Floating-points**
  - **float (32-bit)**
  - **double (64-bit, preferred)**
- **Character: stored as a small integer in computer**
  - **char, unsigned char (8-bit)**
- **Boolean**
  - **bool (true or false, occupies one byte)**

Common Programming Errors

- **Be careful:**

  1/2 gives 0 in C++

  int t;
  1/(t+4) gives 0 in C++

Common Programming Errors

- **Wrong identifiers**
  - **For example,**

    A_{n} a_(n-1) 3tem

    (invalid C++ identifiers)

Valid identifier: consists of only characters:

a to z
A to Z
0 to 9
underscore _
1st character should not be number 0 to 9

Common Programming Errors

- **Forget initializing a variable**
  - **For example,** **double sum;**
    for (int i = 0; i<10; i++) sum += arr[i]; //wrong result

- **Overflow**
  - **We should always ensure that there is no overflow when we are using**
    int, unsigned int, long long, unsigned long long

Common Programming Errors

- **Note that when we are using the double data type, there may be round off errors**

  Example: **int x = pow(10,2);**

  You may not get 100.
  Likely you get 99, since pow(100,2) may return a double value slightly less than 100.

  One solution when the output is int: use int x = round(pow(10,2))
  round() rounds a double to the nearest integer value (still double type);
  we can use this solution here since we know that the output is an integer value.

  But in general, no perfect solution to deal with round-off errors (depending on applications)

Common Programming Errors

- **Be careful of the scope of variable(s)**
  The following code is wrong:
  1) The variable sum inside the for loop is different from the variable sum outside the for loop; 2) the variable sum inside the for loop gets declared (but not initialized) multiple times.

```
int main()
{
    int sum = 0;
    for (int i = 0; i < 100; i++)
    {
        int sum = sum + i;
    }
    cout << sum;
}
```

C++: Arithmetic Operators

- **The common operators: + - * / %**
- **We also have:**

  += *= /= -=

  ++ --

Common Programming Errors

1) There is no exponentiation operator in C++

Do not use a^b to compute a^b in C++
(this error still occurs in our midterm exam)

^ is the bit-wise exclusive OR operator in C++

Common Programming Errors

2) Note that y++ is different from ++y

y++: the value of y is used in that statement first,

then the value of y is increased;

++y: the value of y is increased first,

then the value of y is used in that statement;

3) Do not use: y = y++ or y = ++y

(different compilers give different outputs when you use them.)

C++: Comparison and Logical Operators

Comparison Operators:

< > <= >= == !=

Logical Operators:

&& || !

Order of Operators:

( ) > arithmetic operators > comparison operators > logical operators

if you are uncertain, use ( ) to enforce the order of computation

C++: Control Statements

- **Selection Statements**

  if

  if … else

  switch
- **Loop Statements**

  for

  while

  do … while

Common Programming Errors

```
for (i = 0; i < n; i++) ;
{
    // codes
}
```

```
for (i=0, i < n, i++)
{
    // codes
}
```

```
// for 100 iterations
for (i = 0; i < 101; i++)
      // ….
```

```
for (i = 0; i < i + 3; i++)   // infinite loop
      // ….
```

Common Programming Errors

- **In the condition, assignment operator (=) is used to replace the relational operator (==)**
  - **Serious problem**
    For example:

```
if (i = 7)              // should use    if (i == 7)
```

- **Computed the condition wrongly**

```
if (10 < i < 100)      // should use     if (10 < i && i < 100)
```

Common Programming Errors

- **Do not use braces properly**

```
double x;
double sum = 0;
for (int i = 10; i < 100; i++)
{
    x = sqrt(i);
    sum += x;
}
```

Different

```
double x;
double sum = 0;
for (int i = 10; i < 100; i++)
    x = sqrt(i);
    sum += x;
```

Common Programming Errors

- **Do not use braces properly**
  - **for (int i = 0; i < 10; i++)**
    {
    //…….
    }
    for (int j = 0; j < 20; j++)
    {
    //…..
    }

←→
Different

for (int i = 0; i < 10; i++)
{
//……
for (int j = 0; j < 20; j++)
{
//…..
}
}

C++: Arrays, Vectors, Strings

- **Arrays**
  - **Consists of a sequence of elements of the same data type**
  - **Important for processing multiple data**
  - **In C/C++, the array size is fixed (must be constant at the time of declaration) (our codeblocks allows variable array size, do not use it)**
- **Vectors**
  - **C++ has vectors (C does not have vectors)**
  - **Vector consists of a sequence of elements (the same as array)**
  - **Vector has member functions (vector is a C++ class)**
  - **The vector size can be increased/decreased using member functions:**

    push_back, pop_back, resize, insert, erase

Common Programming Errors

- It is wrong to consider that the index of the first array element is 1
  - Zero based indexing is used in C++ for array, vector, string
  - If there are n elements, the index of the first element is 0;
    the index of the last element is n-1.
- Accessing an element out of the bound of array/vector
  - Results in wrong output or program crash
  - a common error in programming
- Note that resizing one row of a two dimensional vector does not automatically resize the other rows

Common Programming Errors

- **It is wrong to print an array or vector directly (suppose that you want to print out those elements):**

```
int arr[10];                cout << arr;    // wrong
vector<int> vec(5);  cout << vec;   // wrong
```

- **But you can print a string directly**

```
string str = “hello world”;

cout << str;  // it is ok
```

Common Programming Errors

- **It is wrong to return an array in function (you can return a vector/string in function)**

- **It is wrong to assign element to an empty vector directly**

```
vector<int>  vec;
vec[0] = 3;  // wrong; use     vec.push_back(3);
vec[1] = 4;  // wrong; use     vec.push_back(4);
```

Common Programming Errors

- **Wrong member function calls for vector**

```
vector<int> vec(10);

// to erase vec[3]
vec.erase(3);       //wrong; should use  vec.erase(vec.begin()+3);

// to erase vec[3], vec[4], vec[5], vec [6]
vec.erase(3, 4);  // wrong;
                        // use   vec.erase(vec.begin()+3, vec.begin()+7);
```

Common Programming Errors

- **Wrong member function calls for vector (cont.)**

```
vector<int> vec(10);
int x = 3;

// to insert a value at vec[3]
vec.insert(3, x);
//wrong; should use   vec.insert(vec.begin()+3, x);
```

Common Programming Errors

- **Note that some members functions of string cannot be used for vector**

  Example: s.insert(int start, string s1)

- **This year, we did not teach the string erase member function.**
  - **It is to avoid confusion**
  - **You can simply use the vector erase member function for string (with iterator(s))**

```
bar.erase(bar.begin()+2);
bar.erase(bar.begin()+2, bar.begin()+6);
bar.erase(bar.begin()+2, bar.end());
```

C++: Functions

- **Purposes:**
  - **Reuse the code**
  - **Divide a complex task into simpler tasks**
- **Functions**
  - **Declaration**
  - **Definition**
    - **Passing by value by default**
    - **Passing by reference if we want to retain the value of a variable modified in a function**
    - **Passing array to a function is passing by reference by default**
  - **Function call**

Common Programming Errors

- **Data types (including the return type) of function declaration do not match that of function header**

```
void foo(int, int);

int foo(int x, double y)   // do not match the data types in header
{
        // ……
}
```

Common Programming Errors

- **Function call**
  - **Call a function before a function gets declared or defined, wrong.**
  - **We should not include the data type in function call**

```
fun1(int y);          //wrong, should be fun1(y)
fun2(int arr [ ]);    //wrong, should be fun2(arr), here arr is an array
```

Common Programming Errors

- **Separate compilation**
  - **How to compile a program with several source files (.cpp files)**
    - **You should create a project, then add those files of a program to the project (as practiced in lab 4 and 5)**
  - **Do not use**

    #include "yourfile.cpp"
    - **It does not work in separate compilation**
    - **It is poor programming style to include .cpp files**

Common Programming Errors

- **Return**
  - **It is wrong to use**

    return a, b;

    or to use

    return a;

    return b;

- **If you need to return the values of more than one variable/vector/string, you need to use passing by reference**

Common Programming Errors

- **Error in using passing by Reference**
  - **Data type does not match when passing by reference (compilation error)**

```
bool digit_sum(unsigned int n, unsigned int& sum);


int main()
{
   unsigned int n;
   int sum;
   //….

   if (digit_sum(n, sum))   // int sum does not match unsigned int& sum
   //…..
}
```

Common Programming Errors

- **Error in using passing by reference**
  - **Passing by reference is wrongly used for some input parameters (the input parameter gets changed in the function and the modified value get retained)**

```
bool digit_sum(unsigned int& n, unsigned int& sum)
{
        // ….
        n = 1;
        //……
}
```

Infinite loop

```
int main( )
{
   unsigned int n, sum;
   //….

   for (unsigned int i = 0; i < 100; i++)
   {
          if (digit_sum(i, sum))
          //…..
    }
}
```

C++: Input/Output

- **Standard input/output**

  cin >>

  cout <<

- **File input/output**

  ifstream

  ofstream

  Member functions: open, close

  Member operators: >> <<

Common Programming Errors

- **Once we opened a file, we should use the file stream to access the file, not the file name**

```
Example:   ifstream fin;
           fin.open("abc.txt");
           //.....
           fin >> x;          // not   abc.txt >> x;
```

- **Make sure that you know the difference between "getline" and ">>"**
  - **getline(fin, str); => read a whole line into the string str;**
    after executing this function, we access the next line in fin;
  - **fin >> x => extract one data of fin into x; it fails if the data type does not match**
    after executing this operation, we access the next data

Common Programming Errors

- **It is wrong to read an file from a zip file directly in the C++ code:**

  fin.open(C:\\mh1402\\abc.zip\\def.txt)

C++: Pointers and References

- **Pointer**
  - **Contain a memory address**
  - **Declared using the deference operator ***
  - **The memory address of a variable is retrieved using the reference operator &**
    For example: int x; &x
  - **The data at a memory address is retrieved using the deference operator ***
    For example: int* x; *x
- **Reference**
  - **It is the synonym of another variable**
  - **Declared using the reference operator &**
    - **Must be initialized to another variable at the time of declaration**
      For example: int y; int& x = y;

Common Programming Errors

- **Initializing a pointer incorrectly**

```
int* pt;
pt = 5;              // wrong, 5 is likely an invalid address,
                     //        and your program is not allowed to access
```

- **Incorrectly applying the deference operator**

```
int* pt;
int miles = 4;
pt = &45;            //wrong, the value 45 does not have an address
pt = &(miles + 10);  //wrong, the value 14 does not have an address
```

- **Initializing a reference incorrectly**

```
int& y = 45;       //wrong, y must be initialized as another variable
```

C++: Classes

- **Contains member variables, member functions, ….**
  - **Widely used in C++ (vector/string, cin/cout, file input/output)**
  - **Typically the variables are in the private sections**
  - **Functions may be in the private or public sections**
- **An object is an instance of a class**
  - **For example, vector<int> vec(10); // vec is an object**
    // vector is a class

Common Programming Errors

- **It is wrong to give a return type to the constructor**
- **It is wrong NOT to give a return type to other member functions**
- **It is wrong to use the same name for a data member and for a member function in a class**
- **Forget to terminate the class declaration with semicolon**
- **Forget to include the class name and the scope operator (::) in the header when defining a member function outside the class declaration.**

```
int Rectangle::area()
{
        //……
}
```

Semicolon “;”

- **Where to put semicolon “;”**
  - **Semicolon in for loop**

```
for (i = 0; i < 5; i++)  ;      //wrong, “;” indicates an empty statement
{
     // codes in this for loop
 }
```

  - **Semicolon in function header**

```
void foo(int x) ;              //wrong
{
      // codes in this function foo
 }
```

Semicolon ";" (cont.)

- **Semicolon in while loop**

```
do
{
    // codes in this for loop
 }  while (your condition);    //we should have semicolon here
```

- **Semicolon in function declaration**

```
void foo(int ) ;                    //we should have semicolon here
```

- **Semicolon in class declaration**

```
class Rectangle
{
    // ….

};                                  //we should have semicolon here
```

C++: Indent

- **Indent is important for programming**
  - **Easy for you to understand/debug your own code**
  - **Easy for others to understand your code**
- **There are several ways to indent C++ code**
  - **We can follow the Allman style (as illustrated in the lecture notes), and I used the Allman style in almost all the codes in the lectures and lab solutions.**
- **The chance is high that there may be programing error if the codes are not indented properly.**
  - **For example, it would be difficult to find out where a loop ends**

What are not covered in this course?

- **This is a 2AU course with 13hr lecture**
- **We did not learn all the details of C++ programming**
  - **For example, there are many member functions of vector and C++ string**
- **We did not learn those very advanced features of C++**
  - **Pointers**
    - **We learned the basics. Pointers can be quite complicated**
  - **Classes**
    - **We learned the basics.**
    - **Inheritance, Polymorphism are not covered in this course.**
- **In the future, if you encounter something in C++ that are not covered in this course, google or check the C++ books**

Exam

Exam

- **Topics covered**
  - **Control statements (in every question)**
  - **Function (passing by value/reference; separate compilation)**
  - **Array, Vector, String**
  - **File input/output**
  - **Class**
  - **Sort; search**
  - **Random number generation**

Exam

- **Final Exam on 18th April**
  - **open book.**
    You can bring any printed material (books, lecture notes, paper) to the exam.
    You can write anything on the books and lecture notes.
  - **You can copy the codes into the computer before the exam.**

Exam

- **Electronic devices are not allowed in the exam (except calculator)**
- **USB devices are not allowed in the exam**
- **Bring your own draft paper to the exam since it is open book exam (if you need draft paper)**
  - **Do NOT submit any draft paper**
  - **Submit only the codes**
  - **Submit all your codes**
    In the midterm exam, around 10 students did not submit all their solutions

- **Good luck to your exam!**