Questions on 09/18/2023

September 22, 2023

Contents

- Section 5.1 - 5.6, function

In the following, we use function and method interchangeably. Strictly speaking,

- A function is a set of instrunctions concentrating on a task, and a method is a function defined inside a class working with data members in this class.
- A method must be used with an object in the corresponding class. For example, string class has method size(), which returns the number of characters in this string, then we need to use size method with a string object. For example,

```
string str("Hello, world");
cout << str.size() << endl;
```

- In view of employees, you may think main function is the CEO of a company, a function is a contractor, while a method is an employee working only with data associated with a branch (class) of a company.
- Each method/function is independent of each other, it is like each employee is independent, so one method-/function cannot be nested inside another.

Problem 1: Write a function isLeap to find out whether a year is a leap year or not. A leap year is one that can be divided by 4 but not be 100, or a year that can be divided by 400.

Answer: one implementation is as follows.

```
bool isLeap(int year) {
    //since && has higher precedence than ||,
    //we may omit the () around (year % 4 == 0 && year % 100 != 0)
    if ( (year % 4 == 0 && year % 100 != 0) || year % 400 == 0)
       return true;
    else return false;
}
```

The above implementation can be simplified. Observe that, if the condition is true, then return true, otherwise (the condition must be false), then return false. We can return the condition directly. Simplified code is as follows.

```
bool isLeap(int year) {
    return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
}
```

Problem 2: Call function isLeap to find out all leap year from year 2000 to year 2500.

```
int main() {
    for (int year = 2000; year <= 2500; year++)
        if (isLeap(year) == true)
           //isLeap(year) == true can be simplified as isLeap(year)
           cout << year << endl;

    return 0;
}
```

In C++, when main function calls isLeap function, either one of the following approach is used.

Approach 1 Declaration of isLeap function is put ahead of the definition of main function, so that main function can know the signature of isLeap function, that is, what the type of input parameters and what the return type are.

In this approach, **declare** isLeap function first, then **define** main function, finally **define** isLeap function.

We call this declare-use-define approach, which means declare a function first, then use this function in main or other caller function, finally define this function.

In this approach, we see the definition of main function before the definitions of any other function. Since the first line of main function is the entry point of the project, this approach is preferred.

Code can be viewed in link of defining and use isLeap function.

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

    bool isLeap(int year); //declaration of isLeap function

    //definition of main function
    int main() {
        for (int year = 2000; year <= 2500; year++)
            if (isLeap(year))
               cout << year << endl;

        return 0;
    }

    //definition of isLeap function
    bool isLeap(int year) {
        return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
    }
```

Approach 2 The definition of isLeap function is put ahead of the definition of main function, as follows.

```
    bool isLeap(int year) {
        return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
    }

    int main() {
        for (int year = 2000; year <= 2500; year++)
            if (isLeap(year))
               cout << year << endl;

```

```
10          return 0;
11      }
```

Problems for Approach 2 Suppose we have three functionsA, B, C, while main function calls function A, and function A calls function B, which in turn calls C. Then we need to pay attention to the order of definition.

```
definition of function C
definition of function B
definition of function A
definition of main function
```

Suppose we have recursive calling like A calls B, and B calls A. Then Approach 2 will not work, according to the above structure, when B calls A, without the declaration of A putting in front the definition of B, B does not what input parameters and return type of A.

Hence it is better to use Approach 1 as follows.

```
declaration of function A, that is, header of function A followed by ;
declaration of function B
declaration of function C

definition of main function

//with previous declaration, the order of function definition can be arbitrary
definition of function A
definition of function B
definition of function C
```

Questions asked

1. I need a bit more explaining for functions, especially the return and the input.

   I am unsure still how to use void, bool, and others.

   What are the different function types and when do I use them?

   Answer: I will answer all the above questions together. Do you mean input parameter type and/or return type? It depends on which function you are talking about. For example,

   - If a function's return type is void, it does not return anything to its caller.

     Not every method needs a return. For example, clear method of string class erases all the characters of a string object and the corresponding string becomes empty. There is no return in this method.

     See illustrate usage of clear method of string class.

```
1  //illustrate usage of string::clear
2  #include <iostream>
3  #include <string>
4
5  int main () {
6      std::string str("Hello, world");
7      std::cout << "str = \"" << str << "\"" << std::endl;
8         //\" enables to use double quotes inside a pair of "",
```

```
9         //for example, "\"" means the double quotes symbol,
10        //but """ is wrong, since the rightmost " has no matched ".
11
12    str.clear(); //erase contents of the string
13
14    //now str becomes an empty string
15    std::cout << "str = \"" << str << "\"" << std::endl;
16
17    return 0;
18 }
```

- If a function is to calculate the area of a circle, then the input parameter is a radius, which is a double type variable. The area has decimal numbers, so the return type is double. A declaraion is as follows.
  double circleArea(double radius);
- If a function is to return the middle character of a string str if the length of str is odd, or return the middle two characters if the length of str is even.
  The input parameter is a string, the return is a substring, which is a string type. A declaration is as follows.
  string middle(string str);
- If a function is to calculate the result of a base raised to the exponent, then the first parameter is a base, and the second parameter is an exponent, the return is a double type. A declaration is as follows.
  double pow(double base, double exp);
- If a function's return type is bool, it returns either true or false to its caller. This function concentrate on answering questiong like is this year a leap year or a common year, is the first input can be divided by the second input, or if the input number is a prime or not. A declaration is as follows.
  bool isPrime(int num);
- If a function is to find out the sum of all elements of a double-type array, then the input parameter is the name of an array – which implies the initial address of the array – and the size of the array, the return is double type. A declaration is as follows.
  double sum(double arr[], int size);

2. method without parameters?

   Answer: Here are some examples of methods whose input parameter list is empty, aka void.

   - int rand (void);
     Returns a pseudo-random integral number in the range between 0 and `RAND_MAX`.
   - void abort (void);
     Aborts the current process, producing an abnormal program termination.
   - method from string class `size_t size() const;`, where `size_t` is a non-negative integer, and const means the method does not change the string object.
     Returns the length of the string, in terms of bytes.

3. Is void functionName() only for functions that require input?

   Answer: There are different functions, as long as there is no need to return, then the return type is void.

   For example, clear() method of a string class is to erase all characters of the current string. This function does not need to return any feedback to it caller, so the return type is void. Method clear does not need any input parameter.

As another example, function print has the following header. Its functionality is to print str for times time, there is no need to return.

```
void print(string str, int times) {
    for (int i = 0; i < times; i++)
        cout << str;
}
```

4. How to do max and min?

   Answer: you mean how to find max and min in a series of numbers in Task B of Lab 3?

   See Slides 24-26 of slides of loop examples. Code is in finding minimum and maximum of a series of inputs.

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

int main() {
    cout << "Enter numbers: ";
    double num;
    cin >> num;
    //initialization max and min by first number
    double max = num;
    double min = num;

    while (cin >> num) {
        //three mutually exclusive possibilities:
        //(1) num > max
        //(2) num < min
        //(3) min <= num <= max
        if (num > max)
           max = num; //cannot write num = max;
        else if (num < min)
               min = num;
    }

    //outupt the result AFTER all inputs
    //are processed
    cout << "max = " << max << endl;
    cout << "min = " << min << endl;

    return 0;
}
```

5. How do you know when to call a function?

   Answer: it takes practice and experience. Normally, if there is a redundance of codes and the codes are used in different occassions, then we can consider defining a function, put the redundance code there.

   For example, we see the following psuedocode.

   print spaces for numSpaces times

   print asterisks for numAsts times

   Then we may consider to define print function as follows.

```
void print(string str, int times) {
    for (int i = 0; i < times; i++)
```

```
3         cout << str;
4 }
5
6 //To print spaces for numSpaces times, call the above function
7 //print(" ", numSpaces);
8
9 //To print asterisks for numAsts times, call the above function
10 //print("*", numAsts);
```

6. What do most functions begin with types like void, double, etc?

   Answer: type left to the function name is the return type of a function. That is the way a function is defined.

   returnType functionName (parameter-list)

   If there is no return, return type is void.

7. I do not understand certain code lines for Lab 3:
   -The getline() function
   -The fin.ignore line
   -Also, how to get specific data from lines?

   Answer: are you talking about Lab 3?

   ```
   string junk;
   getline(fin, junk);
   ```

   The above code reads a line (may contain spaces) from fstream object fin which is associated with file `"Current_Reservoir_Levels.tsv"`. That is, variable `junk` holds the first line of file with the following contents. Since we do not care the column header information, we save this variable in a variable called junk and do not further process it.

   ```
   Point_time AUGEVolume AUGEASTLEVANALOG  AUGWVOLUME AUGWESTLEVANALOG  ASHREL SICRESVOLUME
       SICRESELEVANALOG  STPALBFLW  RECRESVOLUME  RECRESELEVANALOG  RECREL NICRESVOLUME
       NICRESELEVANALOG  NICNTHFLW  NICSTHFLW  NICCONFLW  EDIRESVOLUME  EDIRESELEVANALOG  EDRNTHFLW
       EDRSTHFLW  EDRCONFLW  WDIRESVOLUME  WDIRESELEVANALOG  WDRFLW
   ```

   After read the data in the first several columns, use `fin.ignore(INT_MAX, '\n');` to ignore the rest of columns.

   When using `fin >> date >> eastSt >> eastEl >> westSt >> westEl`, we read the first five columns, which are date, east storage (eastSt), east elevation (eastEl), west storage (westSt), and west elevation (westEl), then we ignore the rest of columns using `fin.ignore(INT_MAX, '\n');`, print out the information on date and east storage in line `cout << date << " " << eastSt << endl;`, then continue to read the first five columns in the next row by moving back to the while-header. Stop when there is no more data to process. The above process is written by the following codes.

```
1  while(fin >> date >> eastSt >> eastEl >> westSt >> westEl) {
2      // this loop reads the file line-by-line
3      // extracting 5 values on each iteration
4
5      fin.ignore(INT_MAX, '\n'); //skips to the end of line,
6                                 //ignorring the remaining columns
7
8      //process data read. For example, to print the date and East basin storage:
9      cout << date << " " << eastSt << endl;
10 }
```

8. How do you do Project 1B?

   Answer: Please read hints for Project 1B.

9. cin.ignore(): I do not have clean idea aobut its use. I know it is used to ignore user input.

   In Lab 3, we used `fin.ignore(INT_MAX, '\n');` Is it used to ignore the whole line? Will it dump fin all type of variables, like if some inputs are int and some are strings.

   Answer: if you use `fin.ignore(INT_MAX, '\n');` before any `fin >> ...;` statement, then you do not read anything but to ignore the whole current line.

   Assume that you have more than three columns in a row. If you have read three columns from the row using `fin >> variable1 >> variable2 >> variable3;`, followed by `fin.ignore(INT_MAX, '\n');`, then you ignore the rest columns, starting from the fourth column all the way to the end of line.

10. In Lab 3 we took first give variables and ignored rest. What if we need to take first 3 variables and then ignore 3 and then take 2 more.

    Answer: The reading head of file must move from left to right in the a row, once it finishes, move to the leftmost position of the next row. The reading head of file cannot skip a specific number of columns; however it can skip from current position all the way to the end of the current line using `fin.ignore(MAX_INT, '\n');`.

    To get the first 3 columns – column 1, 2, 3, then ignore the next three columns – columns 4, 5, 6, then read next two columns – columns 7 and 8, you may use

```
fin >> variable1 >> variable2 >> variable3
    >> variable4 >> variable5 >> variable6
    >> variable7 >> variable8;
```

    and process only variable1, variable2, variable3, variable7, and variable8, and just do not process the data saved in variable4, variable5, and variable6. You can rename variable4, variable5, variable6 as junk4, junk5, junk6 if you like to indicate that you do not care about those variables.

11. We did some assignments that takes user input and calculate sum. If the user input is like

    2 3 4 Z P Q 10 11

    Z P Q are mistakely put by user. How can computer handle this?

    Answer: if you write code as follows,

```
1 int num;
2 while (cin >> num) {
3     //code to process num is omitted
4 }
```

    Then once Z is entered, `cin >> num` returns false – since `cin >>` operator cannot convert Z to an integer, so the while loop ends. Value Z is not stored in variable num, and rightly so.

12. Is it possible to draw in C++?

    Answer: You can draw in C++. One example is draw a line using library graphics.h. For more complicate figures, you may try matplotlibcpp library.

13. So is function like the title and on writing the method inside of the function. Or method and function are completely different?

    Answer: function and method are both a set of instructions to concentrate on a task. Strictly speaking, a method works with data members in a class and must be associated with an object in that class using dot operator. Please read explanation in function vs. method.

14. I am still having trouble with this topic (function).

    Answer: we will learn more about function and methods in the rest of the semester. Do not worry.

15. Are functions written similarly to how they are written in python?

    Answer: function defintion in C++ and Python are similar.

    Code to calculate area of a circle

```
//Define function to calculate the area of a circle with given radius.
#include <iostream>
#include <cmath>
using namespace std;

double areaCircle(double r);

int main() {
    cout << areaCircle(2) << endl;
    return 0;
}

double areaCircle(double r) {
    return M_PI * r * r;
}
```

    Code to calculate area of a circle in Python is as follows.

```
import math #math.pi
#define function to calculate area of a circle.
def circleArea(r):
    return math.pi * r * r;

def main():
    print(circleArea(2))

if __name__ == '__main__':
   main()
```

16. Can you have multiple functions with the same name?

    Answer: yes. This feature is called method overload.

    - same function name
    - similar functionality
    - different input parameters in terms of number, type, and order. For example, substr method of string class can take either one parameter or two parameters. See the following example, taken from cplusplus website.

      Image you go to an office (analog to a method), one takes an input slot (analog to input parameter), the other take two input slots, when you need to enlist help from that office (analog to call a method), depending on the number of parameters (analog to actual parameters), the system (analog to C++ compiler) knows which office to call.

      Another example is washing machines.

      – A home washing machine takes one input slot, can wash different types of materials, clothes, mats, and so on.

– A commercial washing machine takes two input slots, one take the materials to be washed, the other takes money.
– All washing machines have similar functionality, they all wash materials.
So we may have different wash methods as follows. Suppose Clothesand Mats are different types. Note that we add & after type means pass by reference, that is, when a caller calls a method, it passes the original copy of parameters, so the parameters can be modified (in this example, wash clean) in the callee (in this example, wash method). For more details of pass by reference, please read code link to swap two integers using passing by reference and watch a video videos for passing by reference, starting from 11:50.

```
void wash(Clothes& c); //home washing machine takes in clothes
void wash(Mat& m); //home washing machine takes in mats
void wash(Clothes& c, double money); //commercial washing machine takes in clothes and
    money
void wash(Mat& m, double money); //commercial washing machine takes in mats and money
```

```
// string::substr
#include <iostream>
#include <string>

int main ()
{
  std::string str="We think in generalities, but we live in details.";
                                        // (quoting Alfred N. Whitehead)

  std::string str2 = str.substr (3,5); // "think"

  std::size_t pos = str.find("live");  // position of "live" in str

  std::string str3 = str.substr (pos); // get from "live" to the end

  std::cout << str2 << ' ' << str3 << '\n';

  return 0;
}
```

- It does not matter whether the return type is different or not.

17. Can you go more into details about using fstream functions?

    I need more help with reading a tsv/csv file and calling function.

    Answer: ifstream is to read plain text file, ofstream is to write plain text file, and fstream is to work with binary file like audio and video. In this semester, we only concentrate on ifstream and ofstream.

    Step 1 Open a plain text file to read by instantiating an ifstream object.

    ```
    ifstream fin(fileName); //fileName is a string variable.
    ```

    Step 2 Test the file can be read or not using fail method of fstream class. You may not be able to read the file since the file might be missing or corrupt, or lack of permission.

    ```
    if (fin.fail()) {
       cerr << "The file cannot be opened." << endl;
       exit(1); //leave the program
    }
    ```

Step 3 Use `fin >> variableName;` to read from the file associated with fin. Make sure the data value matches the type of `variableName`.

Step 4 After reading, use `fin.close();` to close the file associated with fin.

To write to a file, the steps are similar.

Step 1 Open a plain text file to write by instantiating an ofstream object.

```
ofstream fout(fileName); //fileName is a string variable.
```

Step 2 Test the file can be write or not using fail method of fstream class. You may not be able to write to the file due to permission.

```
if (fout.fail()) {
   cerr << "The file cannot be opened." << endl;
   exit(1); //leave the program
}
```

Step 3 Use `fout << variableName;` to write to the file associated with fout.

Step 4 After all writing, use `fout.close();` to close the file associated with fout.

Here is an example of reading from and writing to a plain text file. See read from and write to plain text file example.

```
1  //read a plain text file, say grades with the following contents
2  //name exam1 exam2 final
3  //Ann 88   87   86
4  //Bob 77   76   75
5  //Charles 66 65  64
6
7  //generate a file with the following information
8  //name total letter grade
9  //Ann 86.75 B
10 //Bob 75.75 C
11 //Charles 64.75 D
12
13 #include <iostream>
14 #include <fstream> //input and output to plain text file
15 using namespace std;
16
17 //convert numerical grade to letter grade
18 char numericalToLetter(double);
19
20 int main()
21 {
22    ifstream fin("grades.txt");
23
24    if (fin.fail()) { //in case the file cannot be opened
25       cerr << "file cannot be opened" << endl;
26       exit(1);
27    }
28
29    ofstream fout("results.txt");
30
```

```
31      if (fout.fail()) { //in case the file cannot be opened
32          cerr << "file cannot be opened" << endl;
33          exit(1);
34      }
35
36      fout << "name total letter grade" << endl;
37
38      string columnNames;
39      getline(fin, columnNames); //read the first line, no need process
40
41      //Starting from the second line of
42      //text file connected by fin,
43      //each line contains
44      //a string representing name,
45      //three ints representing exam1, exam2,
46      //and final grade respectively.
47      string name;
48      int exam1;
49      int exam2;
50      int finalGrade;
51      double total;
52      while (fin >> name >> exam1 >> exam2 >> finalGrade)
53      {
54          //suppose each exam counts for 25%, written as 0.25,
55          //and the final exam counts for 50%.
56          total = 0.25 * exam1 + 0.25 * exam2 + 0.5 * finalGrade;
57
58          //instead of cout << name << " " << total;
59          //which prints to the console
60          //fout << name << " " << total << endl;
61
62          //Put name and total to fout,
63          //which is connected to
64          //results.txt.
65          //That is, write name, total, and corresponding letter grade,
66          //obtained by numericalToLetter(total),
67          //to the text file associated with fout.
68          fout << name << " " << total << " "
69               << numericalToLetter(total) << endl;
70      }
71
72      fin.close();
73      fout.close();
74      return 0;
75  }
76
77  char numericalToLetter(double score)
78  {
79      char grade;
80      if (score >= 90)
81        grade = 'A'; //cannot write 'A' as "A", since "A" is not
82                     //a char.
83      else if (score >= 80)
84             grade = 'B';
85          else if (score >= 70)
```

```
                grade = 'C';
            else if (score >= 60)
                    grade = 'D';
                else grade = 'F';

    return grade;
}
```

Note when you test in the local machine, you need to create grades.txt in the same directory when you put the source code. Contents of grades.txt is as follows.

```
name exam1 exam2 final
Ann  88    87    86
Bob  77    76    75
Charles 66 65    64
```

18. Do we need to add `#include <cmath>` in the header file when doing math calculation? Also when do we know it's needed in the header file? For example, `#include<string.h>`, why cannot we add it in the int main() code like "string num;" as a type?

    Answer: if we use math function like pow and sqrt, we need to import cmath library in C++. If write a C program, use `include <math.h>`.

    A complete list of functions in math libary is in C++ math library.

    The equivalent of `<string.h>` in C++ is `<string>`. Here is an example of using string in C++ from size method of string class.

```
// string::size
#include <iostream>
#include <string>

int main ()
{
  std::string str ("Test string");
  std::cout << "The size of str is " << str.size() << " bytes.\n";
  return 0;
}
```

19. Can we maybe do a quick review of functions next lecture? I'm honestly still a little lost one it.

    Answer: yes, please read course website. Slides and videos are posted.

20. I am still a little confused about how to get an upside down triangle. I tried to print out a regular triangle and then reverse the order, but I'm not getting the result I want.

    Answer: please work on Task F (Upside-down trapezoid or triangle) in hints for Lab 4. You can also watch video hints for Project 1C, 1D, Labs 4 and 5.

21. Do not really understand Lab 3 even though someone explained it to me.

    Answer: please talk with TAs, lab instructors or me. We would be glad to help.

22. I notice that the syntax is familiar to Java. What are the differences in terms of using C++ vs. Java?

    Answer: Java is similar to C++, with the following main differences.

- In Java, you do not need to define a function before using it as in C++.
- Java has no pointer concept, the details of garbage collection are hidden from users.
- In general, Java runs slower than C++.

  Here is an example to define and test a function to decide whether a year is a leap year or not in Java.

```
public class isLeap {
    static boolean isLeap(int year) {
       return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
    }

    public static void main(String[] args) {
     for (int year = 2000; year <= 2500; year++)
        if (isLeap(year) == true)
           //isLeap(year) == true can be simplified as isLeap(year)
           System.out.println(year);
    }
}
```

23. Where can I see the lecture of today's class?

    Answer: Please check course website.

24. I believe this course is moving along too fast for me personally and have a hard time catching up all the time and it is only the beginning of the course as well.

    Answer: At this point, the difficult part is how to use repetition statements to solve problems. I will give hints on difficult assignments. At the same time, if you have questions, please talk with us.

    - Go to tutoring, which opens from 11:30 AM to 5:30 PM from Monday to Friday when school opens. Location is North Building 1001 B.
    - My office hours are 10:00 - 11:30 AM M, Th at North Building 1001 L.
    - Ask questions in Piazza.

25. Will quizzes be graded with partial credit?

    Answer: yes.

26. Will this class only be taught in C++?

    Answer: yes.

27. How many quizzes and homework we can miss?

    Answer: one quiz and one homework.

28. I would like to spend more time going over the different applications of arrays.

    Answer: starting from 9/28/23, we will introduce array again. An example is to convert 0-9 to English name, you can use nested if-else statment, but you can also use array, the speed is fast, with the expense of memory on arrays. Code is in convert 0-999 to English name.

    Here is an implementation.

```
//leetcode link: https://leetcode.com/problems/integer-to-english-words/
#include <iostream>
#include <cassert>
using namespace std;
```

```
5
6  string int_name(int num);
7
8  //convert 0-9 to names, for num not in this range, return "".
9  string digit_name(int num);
10
11 //convert 10-19 to names, for num not in this range, return "".
12 string teens_name(int num);
13
14 //convert 10, 20, 30, ..., 90 to names, for num not in the range, return "".
15 string tens_name(int num);
16 string zero_to_99_name(int num);
17 string numberToWords(int num);
18
19 int main() {
20     int values[] = {5, 12, 60, 21, 300, 101, 319, 910, 876};
21     string expected[] = {"Five", "Twelve", "Sixty", "Twenty One",
22        "Three Hundred", "One Hundred One",
23        "Three Hundred Nineteen", "Nine Hundred Ten",
24        "Eight Hundred Seventy Six"};
25
26     int size = sizeof(values) / sizeof(values[0]);
27     for (int i = 0; i < size; i++)
28         cout << values[i] << ": "
29              << "\"" + int_name(values[i]) + "\""
30              << endl;
31
32     //test one of 1-9: 5
33     //test one of 11-19: 12
34     //test one of 10, 20, 30, ..., 90: 60
35     //test one of 21-29, 31-39, ..., 91-99: 21
36     //test one of 100, 200, 300, ..., 900: 300
37     //test one of number >= 100 but cannot be divided by 100: 101
38     //test one of number >= 100 but cannot be divided by 100: 319
39     //test one of number >= 100 but can be divided by 100: 300
40     //test one of number >= 100 and number % 10 == 0: 910, 860
41     //test one of number >= 100 but number % 10 != 0 and number % 100 != 0: 876
42     for (int i = 0; i < size; i++)
43         assert(int_name(values[i]) == expected[i]);
44
45     return 0;
46 }
47
48 string digit_name(int num) {
49     string names[] = {"Zero", "One", "Two", "Three",
50         "Four", "Five", "Six", "Seven", "Eight",
51         "Nine", "Ten"};
52
53     if (num < 0 || num > 9)
54       return "";
55     else return names[num];
56 }
57
58 string teens_name(int num) {
59     string names[] = {"Ten", "Eleven", "Twelve", "Thirteen",
```

```
60          "Fourteen", "Fifteen", "Sixteen", "Seventeen",
61          "Eighteen", "Nineteen"};
62
63      if (num < 10 || num > 19)
64         return "";
65      else return names[num % 10];
66  }
67
68  string tens_name(int num) {
69      string names[] = {"", "", "Twenty", "Thirty",
70          "Forty", "Fifty", "Sixty", "Seventy",
71          "Eighty", "Ninety"};
72
73      //only works for 20, ..., 90
74      if (num >= 20 && num <= 90 && num % 10 == 0)
75         return names[num / 10];
76      else return "";
77  }
78
79  string zero_to_99_name(int num) {
80      //not working for negative number
81      if (num < 0 || num > 99)
82         return "";
83
84      if (num <= 9)
85         return digit_name(num);
86      else if (num <= 19)
87             return teens_name(num);
88          else //num >= 20
89               if (num % 10 == 0) // num == 20, 30, ..., 90
90                  return tens_name(num);
91               else return tens_name(num / 10 * 10) + " " +
92                           digit_name(num % 10);
93  }
94
95  string int_name(int num) {
96      if (num < 0 || num > 999)
97         return "";
98
99      if (num < 100)
100        return zero_to_99_name(num);
101     else if (num % 100 == 0)
102            return digit_name(num / 100) + " Hundred";
103     else //num > 100 && num % 100 != 0
104          return digit_name(num / 100) + " Hundred "                          + zero_to_99_name(num %
                100);
105 }
106
107 //convert int from 0 to 2^31 -1
108 string numberToWords(int num) {
109     if (num == 0)
110        return "Zero";
111
112     const int THOUSAND = 1000;
113     string units[] = {"", "Thousand", "Million", "Billion"};
```

```
114
115     string str = "";
116     int ddd; //three-digit tuple
117     int i = 0;
118     string curr;
119     while (num > 0) {
120         ddd = num % THOUSAND;
121         if (ddd != 0) {
122            curr = int_name(ddd);
123            if (str != "")
124               str = curr + " " + units[i] + " " + str;
125            else {
126                  str = curr;
127                  if (i != 0) //units[0] == ""
128                     str += " " + units[i];
129            }
130         }
131
132         //prepare for the next round
133         num /= THOUSAND;
134         i++;
135     }
136
137     return str;
138 }
```

29. Will the readings be necessary to pass the course?

    Answer: yes. If you learn C++ for the first time, it is a good idea to read a textbook.

30. Will we need to make more functions in the future?

    Answer: yes.

31. good sites for my study

    Answer: video: C Programming Tutorial for Beginners from youtube.

    Link: C Programming Tutorial for Beginners

    book: C Programming: A Modern Approach

    book of "C Programming: A Modern Approach".

    If you have more time, go to leetcode.com to work on problems.

32. How will exams be performed? Lab or auditorium? Paper or PC?

    Answer: Midterm and final will be in auditorium, paper-format.

33. When will be the midterm?

    Answer: please read course website. The midterm will be in class on October 16.

34. What is the midterm covering? Will there be a cheat sheet for the exam?

    Answer: the midterm will cover all the materials we have before that day. Please read course website.

    We will have a review on 10/12/23. We will provide a cheat sheet as cheat sheet for the midterm and the final.

35. Can we please make the recitaiton and code review appointment based?

    Answer: it might be difficult to set up navigate, course management system at Hunter, in the mid of semester.