PHYSICS SEMESTER PROJECT

Amir Haytham Salama

Abstract

My project is discussed most of operations that is included in our curriculum. I know that this operation is a bit easy, but making it in a different way is required. So, let us discuss the main operations and how it is different and efficient

At First, in real life projects, there is a common things, one of it, is how to make your program is organized. I mean, every function I wrote to execute required operation, it takes almost 90:100 line of code. So, if the all operations is 9 * 100 line of code or more just in one file, it is not doable, and think about it, if we are in the future want to fix a bug. So, making traditional way, go with your code till you reach the desired line, and fix the bug. Instead, organizing the code in files, that I will explain it about its nature. So, in real life projects, programmers always making a header a files to contain the declaration of functions, structs, classes and any pertinent information. And there is another file include all implementations of these declarations. Thereby, you can look at that like the header file or .h files is considered as an interface and implementation or .cpp files, to tell us how these interfaces is work.

So far, the first thing I do, is how to organize my code before anything to be implemented. Therefore, I organized my project like that. There is two .h files. First one, includes its name MyFunctions.h, which includes all the operations that will be implemented in my program, and this MyFunctions.h, I include it in main, like that `#include "MyFunctions.h",` this line will be in header section to include my own functions. Second .h file that is included in my program Drives DriveFunctions.h, which will have the ability to function view menu a user to choose from, and another function to call take the number of the desired operation, and call it from `MyFunctions.h.` Also, in the main function, we will include `#include"DriveFunctions.h".` Surely, for every .h file, there is .cpp file to execute all the declarations inside it. Also, in gcc compiler provide us whether include many libraries to implement built-in function, there is a library which has all the implementation library, which is `#include<bits/stdc++.h>` it saves some headers includes lines.

For now, we compare the traditional way and the efficient or common used way, which is typically used in real life projects. So, back to the point after organizing my code. Namely, what is operations implemented in my program? And the operation is:

1) OR gate and show truth table
2) AND gate and show truth table
3) NOT gate and show truth table
4) Buffer gate and show truth table
5) XOR gate and show truth table
6) XNOR gate and show truth table
7) NOR gate and show truth table
8) NAND gate and show truth table
9) 1's complement
10) 2's complement
11) Adding two numbers as a binary. Also, as a decimal and display it in binary
12) Bases conversions

Let us discuss the behavior of every function of the previous numeric ones. First, I have to be conscious when the menu will be displayed to the user. And the user will choose specific operation. Let us think if the user see in the console. Choose from [1 … 12], and he was determined to enter string for example. Instead, the integer. Therefore, the programmer should handle this in his program. To make sure that the entered values is right. Thereby, the scenario of handling wrong input data is primarily required. So, utilization built functions with

my code will make a big difference. Note, if my words not completely cleared. Run my code and when the program is displayed try to input any character. Instead, integer value and it will make the point.

Thus, we have discussed in the abstract pages, what I am doing so far to handle some of the process to make it as possible to be efficient process, like I handle it in files and validate from user input to make sure the beginning of the process is completely right. And based on this, discussion of the program operation will be discussed in organized way.

Code flow

The previous scenario is built to be able to make the procedural following process, is completely right.

There is a common things between functions. I mean, in OR, AND, XOR, NOR and NAND gate. The program ask the user to enter two binary strings, and the user will choose the program will process on. But, NOT gate as we know is treated with only one binary string. After all of this operation the result will be displayed in the truth table. Note, input strings should be form same size if you input two binary strings.

```
void ORprocess() {

      string s1, s2;
      cout << "\n" << string(86, '-') << "\n";

      cout << "\n" << string(19, ' ') << string(13, '-') << string(3, '<')
                  << " OR GATE " << string(3, '>') << string(11, '-') << string(15,
                  ' ') << "\n\n";
      cout << string(19, ' ') << "Enter two binary strings from same size  \n\n";

      cout << "Enter first  binary string: ";
      cin >> s1;
      cout << "Enter second binary string: ";
      cin >> s2;
      cout << "\n";
      cout << string(30, ' ') << "| x " << "|" << " y | " << "Result |\n";
      cout << string(30, ' ') << "|---" << "|" << "---|" << "---------|\n";

      for (int i = 0; i < s2.size(); ++i)
            cout << string(30, ' ') << "| " << s1[i] << " | " << s2[i] << " | "
                        << "   " << (s1[i] - '0' | s2[i] - '0') << string(4, ' ')
                        << "|\n";

      cout << "\n" << string(86, '-') << "\n";
}
```

(OR GATE One of the program functions)

Next, also is one of the interestingly operations which is One’s complement, is another process, which is flipping every bit, if I was 0 => 1, and vice versa. So, the program will read just one binary string. And the process will go on.

```
void One_sComplement() {

	string s;

	cout << "\n" << string(86, '-') << "\n";
	cout << "\n" << string(19, ' ') << string(13, '-') << string(3, '<')
	<< " One's Complement " << string(3, '>') << string(11, '-')
	<< string(15, ' ') << "\n\n";
	cout << "Enter the binary string to get its 1'S complement : ";
	cin >> s;
	cout << "Result : ";
	for (int i = 0; i < s.size(); ++i)
		cout << !(s[i] - '0');
	cout << "\n" << string(86, '-') << "\n";}
```

(In previous code. Like the idea I said, I do it as a code, which flipping every digit on the binary string)

Furthermore, another one of the interestingly operations which is two’s complement, is another process, which is first making One’s complement. After that, add one. But we have to notice when we think about two’s complement, which is negative number from a positive one. So, the two’s complement of 2 is -2. Also, there is an observation which is when we want to obtain two’s complement for this 100010 it will be 011110. I mean, that we can do it by. Find the first one from right. Ok!, after that, flip digits values which to the left of this 1. And you will obtain result manually.

```
void two_sComplement() {

      cout << "\n" << string(86, '-') << "\n";
      cout << "\n" << string(19, ' ') << string(13, '-') << string(3, '<')
                  << " two's Complement " << string(3, '>') << string(11, '')
                  << string(15, ' ') << "\n\n";
      cout << "Enter The binary string to get 2's complement: ";
      string s2;
      cin >> s2;
      int ind = s2.find_last_of('1');
      for (int i = 0; i != ind; ++i) {
            s2[i] = (!(s2[i] - '0')) + '0';//casting the character to integer
                                          //after that, flip its digit
                                          //after that,return it string again
      }
      cout << "Result : " << s2;
      cout << "\n" << string(86, '-') << "\n";

}
```

Also, previous code will ask (not the user) about the first one from the right, once it got. So, the digits from index zero to the index which is before the last one, it will be flipped

Besides, Adding two numbers in its decimal representations and in its binary representation, and. In both cases, if you choose to enter the decimal values, the program will display the binary representations of every decimal values and the sum in decimal and binary representations. And vice versa

```
void AddingTwoNumbers() {
	string s1, s2;
	cout << "\n" << string(86, '-') << "\n";
	cout << "\n" << string(19, ' ') << string(13, '-') << string(3, '<')
			<< " Adding Two Numbers " << string(3, '>') << string(11, '-')
			<< string(15, ' ') << "\n\n";
	Cout << "if you want to enter them as binary string representation choose 1. or, 
decimal number choose 2 : ";
	int n_;
	cin >> n_;
	if (n_ == 1) {

		cout << "Enter first  binary string: ";
		cin >> s1;
		cout << "Enter second binary string: ";
		cin >> s2;
		auto TestSize = [&](string &s1_,string &s2_) {
			if ( s1_.size() < s2_.size() ) {
				int sz = int( s2_.size() - s1_.size() );
				for(int i = 1; i <= sz; ++i)
				s1_.insert(s1_.begin(),'0');
			}
			else if ( s1_.size() > s2_.size() ) {
				int sz = int( s1_.size() - s2_.size() );
				for(int i = 1; i <= sz; ++i)
				s2_.insert(s2_.begin(),'0');
			}
		};
		auto convert = [&](string s1_) {
			reverse(all(s1_));
			int sz = s1_.size();
			int no = 0LL;
			for(int i = 0; i < sz; ++i) {
				no+=(1LL<<i)*(s1_[i]-'0');
			}
			return no;
		};
		auto converttostr = [&](long long no) {
			string s_ = "";
			while(no) {
				s_.insert(s_.begin(),( no%2LL +'0'));
				no/=2LL;
			}
			//reverse(all(s));
			return s_;
		};
		TestSize(s1, s2);
		cout << convert(s1) << " + " << convert(s2) << " = " << convert(s1)
				+ convert(s2) << "\n";
		auto DisplayInSameSize = [&]( string &_ ) {
			if(_.size() == s1.size())///since, s1.size()==s2.size(), doesn't matter 
s1 or s2
			return _;
			else if( _.size() < s1.size() ) {
				for(int i = 1; i <= ( s1.size() - _.size() ); ++i)
				_.insert(_.begin(),'0');
```

```
            }
            else if( _.size() > s1.size() ) {
                  for(int i = 1; i <= ( _.size() - s1.size() ); ++i)
                  _.insert(_.begin(),'0');
            }
            return _;
      };
      string tmp = converttostr(convert(s1) + convert(s2));
      cout << s1 << " + " << s2 << " = " << DisplayInSameSize(tmp) << "\n";

} else if (n_ == 2) {
      auto TestSize = [&](string &s1,string &s2) {
            if ( s1.size() < s2.size() ) {
                  int sz = int( s2.size() - s1.size() );
                  for(int i = 1; i <= sz; ++i)
                  s1.insert(s1.begin(),'0');
            }
            else if ( s1.size() > s2.size() ) {
                  int sz = int( s1.size() - s2.size() );
                  for(int i = 1; i <= sz; ++i)
                  s2.insert(s2.begin(),'0');
            }
      };
      auto convert = [&](string s1) {
            reverse(all(s1));
            int sz = s1.size();
            long long no = 0LL;
            for(int i = 0; i < sz; ++i) {
                  no+=( (1LL<<i)*(s1[i]-'0') );
            }
            return no;
      };
      auto converttostr = [&](long long no) {
            string s = "";
            while(no) {
                  s.insert(s.begin(),( no%2LL +'0'));
                  no/=2LL;
            }
            return s;
      };
      long long n1, n2;
      cout << "Enter first  decimal number: ";
      cin >> n1;
      cout << "Enter second decimal number: ";
      cin >> n2;
      string s1 = converttostr(n1), s2 = converttostr(n2);
      TestSize(s1, s2);
      auto DisplayInSameSize = [&]( string &_ ) {
            if(_.size() == s1.size())///since, s1.size()==s2.size(), doesn't matter
s1 or s2
            return _;
            else if( _.size() < s1.size() ) {
                  int sz = int( s1.size() - _.size() );
                  for(int i = 1; i <= sz; ++i)
                  _.insert(_.begin(),'0');
            }
            else if( _.size() > s1.size() ) {
                  int sz = int( _.size() - s1.size() );
                  for(int i = 1; i <= sz; ++i)
                  _.insert(_.begin(),'0');
            }
```

```
                    return _;
                };
                cout << n1 << " + " << n2 << " = " << n1 + n2 << "\n";
                string tmp = converttostr(n1 + n2);
                cout << s1 << " + " << s2 << " = " << DisplayInSameSize(tmp) << "\n";
        }
        cout << "\n" << string(86, '-') << "\n";
}
```

In previous code. So, the function is divided into two parts, if the user choose the binary representation, so the flow will go on the condition scope `if (n_ == 1),` there is a function on the fly which is used inside this scope, which wrote the function on, so there is one like `TestSize,` Assume this scenario you asked user to enter two binary strings, so he does the following 100010, 0101. However, they are not from same size, by using `insert` built-in which takes position and the value. Thus, zeros will be added at the left of the string to make the addition valid. Also, using `convert` function, to convert binary string into decimal value to display in output. In addition to, the function `DisplayInSameSize ()`, which takes one parameter string and display the resulting addition binary string in the same size of input data. Similarly, if the user enter `2,` the flow will go on in the condition scope of `if (n_ == 2),` and the user will be asked to enter two decimal number, same functions will be used again, but in different order. Note, you can enter two positive decimal values or two negative decimal values. Check this in code

9 => 1001

-2 => 1110 (Note this form in two's complement form)

Sum => 7 => 0111

Another Test case

-9 => 0111 (Note this form in two's complement form)

2 => 0010

Sum => -7 => 1001 (Note this form in two's complement form)

Finally, showing bases. Simply, the program will ask the user to enter number that will be shown in another bases forms. Like if entered number is decimal, it will be shown in binary, octal and hexadecimal. Using also built-in functions it makes a big difference. To avoid dozens of bugs and the program will show the flow.

```
#include "MyFunctons.h"
#include <bits/stdc++.h>
#define all(v) v.begin(),v.end()

using namespace std;

int BasesMenu() {

	cout << "1) Binary\n";
	cout << "2) Decimal\n";
	cout << "3) Hexadecimal\n";
	cout << "4) Octal\n";
	cout << string(28, ' ') << "Please choose from [1 ... 4] : ";
	int iChoice_;
	cin >> iChoice_;
	if (cin.fail()) {
		cout << "\n" << string(33, ' ') << string(4, '-')
		        << "<< Wrong Input >>" << string(4, ' -') << "\n";
		cin.clear();
		cin.ignore(10000, '\n');
		return BasesMenu();
	} else
		return iChoice_;
}

void basesConversions() {

	auto converttostr = [&](long long no) {
		string s_ = "";
		while(no) {
			s_.insert(s_.begin(),( no%2LL +'0'));
			no/=2LL;
		}
		return (s_ == "" ? "0" : s_ );
	};

	auto convert = [&](string s1_) {
		reverse(all(s1_));
		int sz = s1_.size();
		long long no = 0LL;
		for(long long i = 0; i < sz; ++i) {
			no+=(1LL<<i)*(s1_[i]-'0');
		}
		return no;
	};

	auto converttodec_hex = [&](string hexa) {
		reverse(all(hexa));
		int sz = (int)hexa.size();
		long long tmp = 0LL;
		long long no_ = 0LL;
		for(int i = 0; i < sz; ++i) {
			if ( hexa[i] >= 'a' && hexa[i] <= 'f' ) {
				tmp = (hexa[i]-'a')+10;
			}
			else if ( hexa[i] >= 'A' && hexa[i] <= 'F' ) {
```

```
                        tmp = (hexa[i]-'A')+10;
                }
                else if ( hexa[i] >= '0' && hexa[i] <= '9' ) {
                        tmp = (hexa[i]-'0');
                }
                no_ += pow(16,i)*(tmp);
        }
        return no_;
};

auto testHex = [](string Hex_) {
        string ValidValues = {"0123456789aAbBcCdDeEfFxX"};
        int sz = (int)Hex_.size();
        for( int i = 0; i < sz; ++i) {
                if ( find( all ( ValidValues ) , Hex_[i] ) == ValidValues.end() ) {
                        cout << "Not allowed number\n";
                        printf("value of Hexadecimal value in binary  : Not Valid\n");
                        printf("value of Hexadecimal value in octal   : Not Valid\n");
                        printf("value of Hexadecimal value in decimal : Not Valid\n");
                        return 0;
                }
        }
};

auto testOct = [](string oct_) {
        string ValidValues = {"01234567"};
        int sz = (int)oct_.size();
        for( int i = 0; i < sz; ++i) {
                if ( find( all ( ValidValues ) , oct_[i] ) == ValidValues.end() ) {
                        cout << "You have to know that the most greater bit in octal
representation doesn't exceed 7 \n";
                        printf("value of octal value in binary      : Not Valid\n");
                        printf("value of octal value in decimal     : Not Valid\n");
                        printf("value of octal value in Hexadecimal: Not Valid\n");
                        return 0;
                }
        }
};

auto converttodec_oct= [&](string oct) {
        reverse(all(oct));
        int sz = (int)oct.size();
        long long tmp = 0LL;
        long long no_ = 0LL;
        testOct(oct);
        for(int i = 0; i < sz; ++i) {
                if ( oct[i] >= '0' && oct[i] <= '7' ) {
                        tmp = (oct[i]-'0');
                }
                no_ += pow(8,i)*(tmp);
        }
        return no_;
};

int n = BasesMenu();
switch (n) {
case 1: {
        cout << "\nEnter string binary value : ";
        string s;
        cin >> s;
        long long no = convert(s);
```

```
            printf("value of binary value in decimal    : %d\n", no);
            printf("value of binary value in octal      : %o\n", no);
            printf("value of binary value in Hexadecimal: %X\n", no);
            break;
        }
        case 2: {
            cout << "\nEnter decimal value : ";
            long long no;
            cin >> no;
            string s = converttostr(no);
            printf("value of decimal value in binary     : %s\n", s.c_str());
            printf("value of decimal value in octal      : %o\n", no);
            printf("value of decimal value in Hexadecimal: %X\n", no);
            break;
        }

        case 3: {
            cout << "\nEnter hexadecimal value : ";
            string no;
            cin >> no; ///Reading as Hexadecimal value
            if (!testHex(no)) {
                break;
            }
            long long no_dec = converttodec_hex(no);
            string s = converttostr(no_dec);
            printf("value of decimal value in binary     : %s\n", s.c_str());
            printf("value of hexadecimal value in octal  : %o\n", no_dec);
            printf("value of hexadecimal value in decimal: %d\n", no_dec);
            break;
        }

        case 4: {
            cout << "\nEnter octal value : ";
            string no;
            cin >> no; ///Reading as octal value
            if (!testOct(no)) {
                break;
            }
            long long no_dec = converttodec_oct(no);
            string s = converttostr(no_dec);
            printf("value of octal value in binary      : %s\n", s.c_str());
            printf("value of octal value in hexadecimal : %X\n", no_dec);
            printf("value of octal value in decimal     : %d\n", no_dec);
            break;
        }

        default:
            cout << "\nWrong Choice bases Conversions\n";

        }

}

void getBasesReady() {

        cout << "\n" << string(86, '-') << "\n";

        cout << "\n" << string(28, ' ') << string(4, '-') << string(3, '<')
                << " Bases Conversions " << string(3, '>') << string(2, '-')
                << string(15, ' ') << "\n\n";
        cout << string(28, ' ') << "From any base you want to start \n\n";
        basesConversions();
        cout << "\n" << string(86, '-') << "\n";

}

void driveBasesConversions() {
        getBasesReady();
}
```

To sum up, the final part consists of four parts; displaying, choosing, verifying and executing. The previous code has three method. First, display the menu, that the user will choose the base, that he want to start from, and the program validate from that, For example, we have four bases to makeuser choose from. So, if the program asked him to enter a number from 1: 4, and the user misbehaved, and choose another thing. Thereby, the program has to warn him from his wrong choice. Moreover, the wrong input also considered. I mean, we talk right now about validation of choosing specific number from 1:4. Think with me, what is going happen, if the user enter string instead of number. Also, have to be warned about this misbehaving. After checking validation from input. There's also on the hexadecimal and octal values. For example, if the user entered in hexadecimal base like "J". So what is this? , or in octal values, the user entered a digit, which is greater than 7. All these validation, I handle it in the program. So, after displaying, validation, choosing the desired operation, doing this in ordered steps, in the functions. So, at the above function is to run the two other functions which is pertinent to the base conversion function.

Try this in valid input in hexadecimal 0xfk

Try this in valid input in octal 01239

Eventually, after discussing in details the operation functions. The time come to list the operations, and make the user decide where he want to go. So, our last function is to derive the operations.

```
int ViewMenu() {
	cout << "\n" << string(86, '-') << "\n";
	cout << string(24, ' ') << "Choose any option to start with :\n\n";
	cout << "1)  OR gate and show truth table\n";
	cout << "2)  AND gate and show truth table\n";
	cout << "3)  NOT gate and show truth table\n";
	cout << "4)  Buffer gate and show truth table\n";
	cout << "5)  XOR gate and show truth table\n";
	cout << "6)  XNOR gate and show truth table\n";
	cout << "7)  NOR gate and show truth table\n";
	cout << "8)  NAND gate and show truth table \n";
	cout << "9)  1's complement \n";
	cout << "10) 2's complement \n";
	cout << "11) Adding two numbers as a binary. Also, as a decimal and display it in 
binary \n";
	cout << "12) Bases conversions \n";
	cout << "\n" << string(86, '-') << "\n";
	cout << "\nEnter a number from [1 ... 12] to start the process : ";
	int iChoice = 0;
	cin >> iChoice;
	if (cin.fail()) {

		cout << "\n\n" << string(20, ' ') << string(10, '*') << string(3, '<')
				<< " Warning " << string(3, '>') << string(10, '*') << string(
				15, ' ') << "\n\n";
		cout << string(3, ' ')
				<< "Please, the the data that you entered isn't a desired data 
type.\n";
		cout << string(3, ' ')
				<< "So, Iam asking you to enter integer to process. Check it 
again.\n\n";
		cout << string(20, ' ') << string(35, '*') << string(15, ' ') << "\n\n";

		cin.clear();
		cin.ignore(10000, '\n');
		return ViewMenu();
	}
	return iChoice;
}
void DriveOperations() {
	char ch;
	bool bCont = 1;
	while (bCont) {
		int iChoice = ViewMenu();
		switch (iChoice) {
		case 1:
			ORprocess();
			break;
		case 2:
			ANDprocess();
			break;
		case 3:
			NOTprocess();
			break;
		case 4:
			Bufferprocess();
			break;
```

```
            case 5:
                  XORprocess();
                  break;
            case 6:
                  XNORprocess();
                  ;
                  break;
            case 7:
                  NORprocess();
                  break;
            case 8:
                  NANDprocess();
                  break;
            case 9:
                  One_sComplement();
                  break;
            case 10:
                  two_sComplement();
                  break;
            case 11:
                  AddingTwoNumbers();
                  break;
            case 12:
                  driveBasesConversions();
                  break;
            default:
                  cout << "\n\n" << string(20, ' ') << string(10, '*') << string(3,
                              '<') << " Wrong choice " << string(3, '>') << string(10,
                              '*') << string(15, ' ') << "\n\n" << string(3, ' ')
                              << "So, Iam asking you to choose from [ 1 .. 12 ]. Check it
again.\n";

            }
            //cout << string(20,' ')<<string(35,'*')<<string(15,' ')<<"\n\n";
            ///Asking user if he want to complete in the process
            cout << string(3, ' ')
                        << "if you want to close the program press [ N or n ] .
otherwise, press any key : ";
            cin >> ch;
            cout << "\n" << string(20, ' ') << string(40, '*') << string(15, ' ')
                        << "\n\n";
            if (ch == 'N' || ch == 'n')
                  bCont = 0;
            else {
                  bCont = 1;
            }
      }
}
```

Note to this line `if (cin.fail())`, what does this line do so far?, like I said, I want to validate from the user input . So, i asked him to enter a number from 1 : 12, think about if he enter string for example. So, this line is executed till user enter a number. You can try this. Open the program, and enter am for example and see.

Thinking strategy

Always, we have to think about how to handle dozens of bugs, which maybe cause the program crashes, and recognizing the nature of the user environment. I mean, the user maybe you asked him in the program to enter just number. However, he entered string for example, or entered a character. So, the programmer have to think how to handle this, if the user determined to do it. Therefore, making your code to be accurate as possible as you can

In addition to, the complexity of the program, or Big O notation, should be handled as possible as we can. Trying to recognizing built-in functions, and implement them in our program to reduce the complexity of the program. Another benefit from built-in function, is reducing a lot of code lines in the program, to be readable and much more efficient.

As we know, there is in data structures vector, string, queue, stack, etc. but also these are fully implemented in c++ language, to be ready for use. To make words clear, there is in c++ Lang class, which is called `string`, this class is fully implemented and has some powerful pertinent functions, like insert, that takes two parameter (position, the value), `s_.insert(s_.begin(),( no%2LL+'0'));`. Also, reverse function, that takes two parameter too, (iterator at the beginning, iterator at the beginning), to reverse the string content, `reverse(s.begin(),s.end());`. Besides, that string class not limited with the specific size, but in array of characters, you are limited in array of characters. Also, you can called it as dynamic array of characters. Another pretty built-in function, which is string (number, character), `string(4,'0');`, this function takes two parameter. First, is the number of repeated times to repeat a specific character number of times. So, using a convenient data structure. Make the e program don't waste time.

Also, there is another powerful and too much awesome thing, which is function on the fly, and the definition for this function is writing the function, but inside the main function. Normally, we always write functions in the global area, and call it the main. But the new in the function on the fly is writing all the function components inside the main function, which can be like the following:

```
auto converttostr = [&](long long no) {
	string s_ = "";
	while(no) {
		s_.insert(s_.begin(),( no%2LL +'0'));
		no/=2LL;
	}
	return s_;
};
```

Note, this function is written inside another function, which is also included in my program. Also, the idea of the function on the fly, if the programmer write a function, while he writes the declaration of this function, he remembers that he needs to get another thing, but the same scope and not to go out of this. In addition, this function so far takes a numbers and convert it into binary representation and insert its values in `string`, return its value.

About the code, all the code is mine. At the end, feel free to run my code, and if my words and explanation not clear, I can come to your office, and discuss this code.