

**class Procedure Reality test report tell**

**Course Name: C Language Programming**

**Professional class: Computer Science and Technology (school transfer) 1 601 class**

**student ID: U201612696**

**Name: Chen Haorui**

**mentor: Gan Zaobin**

**Date of report: 2017.6.8**

**School of Computer Science and Technology**

**Table of contents**

[1 Expression and standard input and output experiment](#_Toc484681458)  [3](#_Toc484681458)

[1.1 Required question](#_Toc484681459)  [3](#_Toc484681459)

[1.2 Self-made questions](#_Toc484681460)  [8](#_Toc484681460)

[1.3 Summary](#_Toc484681461)  [8](#_Toc484681461)

[2 Process Control Experiment](#_Toc484681462)  [10](#_Toc484681462)

[2.1 Required questions](#_Toc484681463)  [10](#_Toc484681463)

[2.2 Optional Question](#_Toc484681464)  [18](#_Toc484681464)

[2.3 Self-made questions](#_Toc484681465)  [19](#_Toc484681465)

[2.4 Summary](#_Toc484681466)  [21](#_Toc484681466)

[3 Function and Program Structure Experiment](#_Toc484681467)  [22](#_Toc484681467)

[3.1 Compulsory questions](#_Toc484681468)  [22](#_Toc484681468)

[3.2 Optional questions](#_Toc484681469)  [28](#_Toc484681469)

[3.3 Self-made questions](#_Toc484681470)  [29](#_Toc484681470)

[3.4 Summary](#_Toc484681471)  [30](#_Toc484681471)

[4 Compilation preprocessing experiment](#_Toc484681472)  [32](#_Toc484681472)

[4.1 Compulsory questions](#_Toc484681473)  [32](#_Toc484681473)

[4.2 Self-made questions](#_Toc484681474)  [39](#_Toc484681474)

[4.3 Summary](#_Toc484681475)  [40](#_Toc484681475)

[5 Array Experiments](#_Toc484681476)  [41](#_Toc484681476)

[5.1 Compulsory questions](#_Toc484681477)  [41](#_Toc484681477)

[5.2 Optional questions](#_Toc484681478)  [53](#_Toc484681478)

[5.3 Self-made questions](#_Toc484681479)  [58](#_Toc484681479)

[5.4 Summary](#_Toc484681480)  [60](#_Toc484681480)

[6 pointer experiment](#_Toc484681481)  [62](#_Toc484681481)

[6.1 Compulsory questions](#_Toc484681482)  [62](#_Toc484681482)

[6.2 Optional questions](#_Toc484681483)  [74](#_Toc484681483)

[6.3 Self-made questions](#_Toc484681484)  [80](#_Toc484681484)

[6.4 Summary](#_Toc484681485)  [81](#_Toc484681485)

[7 Structure and Joint Experiments](#_Toc484681486)  [83](#_Toc484681486)

[7.1 Compulsory questions](#_Toc484681487)  [83](#_Toc484681487)

[7.2 Optional questions](#_Toc484681488)  [94](#_Toc484681488)

[7.3 Self-made questions](#_Toc484681489)  [94](#_Toc484681489)

[7.4 Summary](#_Toc484681490)  [94](#_Toc484681490)

[8 File Experiment](#_Toc484681491)  [96](#_Toc484681491)

[8.1 Compulsory questions](#_Toc484681492)  [96](#_Toc484681492)

[8.2 Self-made questions](#_Toc484681493)  [100](#_Toc484681493)

[8.3 Summary](#_Toc484681494)  [100](#_Toc484681494)

[References](#_Toc484681495)  [101](#_Toc484681495)

# 1 expression and standard input and output experiments

## 1.1 Required questions

⑴ Program error correction: 1 question

C language program routine is given below to accomplish the following tasks:

( 1 ) Input Fahrenheit temperature f , convert it to Celsius temperature c and output;

( 2 ) Input the radius value r of the circle, calculate and output the area s of the circle;

( 3 ) Input short integers k and p, take the high byte of k as the low byte of the result, and the high byte of p as the high byte of the result, form a new integer, and then output it;

There are several syntax and logic errors in this example program. It is required to refer to the steps of 1.3 and 1.4 to debug and modify the following program so that it can correctly complete the specified task.

1 #include <stdio.h>

2 #define PI 3.14159;

3 void main( void )

4 {

5 int f ;

6 short p, k ;

7 double c , r , s ;

8

9 /\* for task 1 \*/

10 printf(“Input Fahrenheit:” );

11 scanf(“%d”, f ) ;

12c = 5/9\*(f-32) ;

13 printf( "\n %d (F) = % .2f(C)\n\n ", f, c ) ;

14

15 /\* for task 2 \*/

16 printf("input the radius r:");

17 scanf("%f", &r);

18 s = PI \* r \* r;

19 printf("\nThe acreage is % .2f\n\n",&s);

20

21 /\* for task 3 \*/

22 printf("input hex int k, p :");

23 scanf("%x %x", &k, &p );

24 newint = (p&0xff00)|(k&0xff00)<<8;

25 printf("new int = %x\n\n",newint);

26

27}

**answer:**

( 1 ) Error modification:

1) There cannot be a semicolon after the definition of the symbolic constant on line 2, and the correct form is :

#define PI 3.14159

2) void and main in line 3 are not separated , the correct form is:

void main( void )

3) The value of r in line 11 has not been modified , and the correct form is:

scanf(“%d”,&f ) ;

4 ) The division in line 12 is wrong, the correct form is:

c = 5.0/9\*(f-32) ;

5) The new int on line 24 is not defined , the correct form is:

(line 6) short p, k , newint ;

6) The input type on line 17 is wrong , and the correct form is :

scanf("%lf", &r);

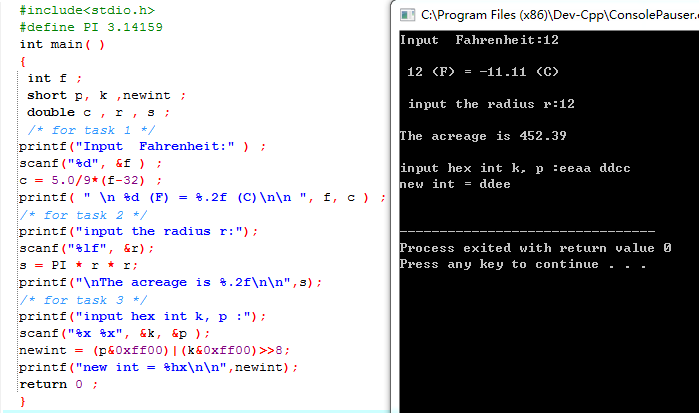
7) The expression in line 24 is wrong, the correct form is :

newint = (p&0xff00)|(k&0xff00) >> 8;

8) The output format on line 25 is wrong, the correct form is :

printf("new int = %hx\n\n",newint);

( 2 ) Running result after error modification:



(2) Program modification and replacement: 1 question

The following program uses the commonly used intermediate variable method to exchange two numbers, please use the exchange method without the third variable instead.

#include <stdio.h>

void main( )

{

int a, b, t;

printf("Input two integers:");

scanf("%d %d",&a,&b);

t=a, a=b, b=t;

prinf("\na=%d,b=%d",a,b);

}

**answer :**

The replaced program looks like this :

#include <stdio.h>

void main( )

{

int a, b, t;

printf("Input two integers:");

scanf("%d %d",&a,&b);

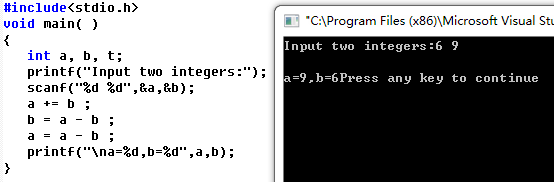
a += b ;

b = a - b ;

a = a - b;

printf("\na=%d,b=%d",a,b);

}



(3) Programming: 2 questions

1. Write a program, input unsigned short integer x , m, n ( 0 ≤ m ≤ 15, 1 ≤ n ≤ 16- m) , and take out the n bits of x from the mth bit to the left (m from right to left numbered from 0 to 15 ), and align it to the left end (the 15th ).

**answer:**

1) Problem-solving ideas:

1. Input x , m , n , in order to analyze the test results conveniently, the input of x adopts hexadecimal

2. Make logic ruler:

2.1 Design ideas : the digits to be reserved correspond to 1 in the logic scale , and the rest correspond to 0;

2.2 0 left shift ( n-1 ) bit, inversion, that is, n bit 1; then left shift m bit , generate m bit 0;

3. Use the logic ruler to filter the digits of the target number and then shift to the left , and the number of left shifts is determined to be 16 – m - n ;

4. Output the result , end

2 ) Program List

# include <stdio.h>

int main()

{

int i , a , j ;

unsigned short x , m , n , mask ;

scanf ( "%d" , &a ) ;

for ( i = 0 ; i < a ; i ++ ) /\* Determine the number of data input groups \*/

{

mask = 1;

scanf ( "%hu %hu %hu" , &x , &m , &n ) ;

if ( m <= 0 || m >= 15 ) /\* judging whether the input of m and n meets the conditions\*/

{

printf ("incorrect input of m\n") ;

}

if ( n <= 1 || n >= ( 16 - m ) )

{

printf ("incorrect input of n\n") ;

}

else

{

mask = ( ~ ( 0 << ( n - 1 ) ) ) << m ; /\*Set the target number to 1 on the logic scale\*/

x &= mask ;

x <<= 15 - m - n + 1 ; /\*The selected digits are shifted to the left\*/

printf ( "%hu \n ", x ) ;

}

}

}

3 ) Test

(a) Test data:

Describe the method for selecting test data. . . As shown in Table 1-1.

Table 1-1 Test data of programming question 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test  Example | program input | | | theoretical results |  |
| x | m | N |
| use case 1 | 0000 0010 0010 1011 (555) | 2 | 5 | Calculation result 0101 0000 0000 0000 is 20480 |
| use case 2 | 0000 0000 0000 0001 (1) | 17 | 1 | m , n are both wrong |
| use case 3 | 0001 0000 1010 0001 (4257) | 11 | 15 | error \_ |

(b) The running results of the corresponding test test cases are shown in Figure 1 .

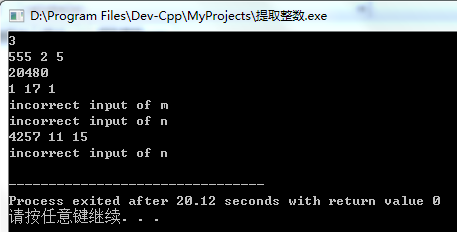


Figure 1 The running results of the test case of programming question 1

It shows that the above operation results are consistent with the theoretical analysis , which verifies the correctness of the program.

2. The IP address is usually 4 small integers separated by periods, such as 32.55.1.102 . These addresses are represented in the machine as unsigned long integers. Write a program that reads in a 32- bit Internet IP address in machine-stored form , decodes it, and outputs it in the usual period-separated 4 -part form. .

**answer:**

1) Problem-solving ideas:

1. Input data number n ;

2. Store the decoded four unsigned integers in an array . Since the decoding is performed in bytes, the array type uses unsigned char will suffice .

3. Since the output low byte is decoded in order from low to high , the mathematical expression of the left shift number of the original integer can be obtained as ( 4 - 1 - j ) \* 8 , where for portability , 4 Use sizeof ( unsigned long ) instead; 8 is replaced by byte length CHAR\_BIT .

4. Shift right by 8\*3 bits . Likewise, 8 is replaced with byte length CHAR\_BIT.

5. Output the numbers in the array in order ;

6. Loop until the number of output groups reaches n ;

6. End .

2 ) Program List

# include <stdio.h>

# include <limits.h>

const int UL = sizeof ( unsigned long ) ;

int main()

{

int i , j , n ;

unsigned long origin ;

scanf ( "%d" , &n ) ; /\* Set the number of input groups\*/

for ( i = 0 ; i < n ; i ++ )

{

unsigned short s[j] = {0};

scanf ( "%lu" , &origin ) ;

for ( j = 0 ; j < 4 ; j ++ )

{

s[j] = ( ( origin << ( UL - 1 - j ) \* CHAR\_BIT ) >> ( ( UL - 1 ) \* CHAR\_BIT ) ) ;

/\* Screen and keep the 4 bytes of the original number in order from low to high and store them in the array. See the question for details on the implementation method\*/

}

for ( j = 0 ; j < 3 ; j ++ )

{

printf ( "%d." , s[j] ) ;

/ \* The problem of reverse order output of each byte of the original number has been considered when screening the digits , and only sequential output is required here \*/

}

printf ( "%d\n" , s[3] ) ;

}

return 0 ;

} 3 ) Test

(a) Test data:

Table 1-1 Test data of programming question 3

|  |  |  |  |
| --- | --- | --- | --- |
| test  Example | program input | theoretical results |  |
| origin \_ |
| use case 1 | 134744072 | 8.8.8.8 |
| use case 2 | 676879571 | 211.92.88.40 |
| use case 3 | 16885952 | 192.168.1.1 |
| use case 4 | 16843009 | 1.1.1.1 |  |
| use case 5 | 26975347 | 115.156.155.1 \_ |  |

(b) The running results of the corresponding test test cases are shown in Figure 2.

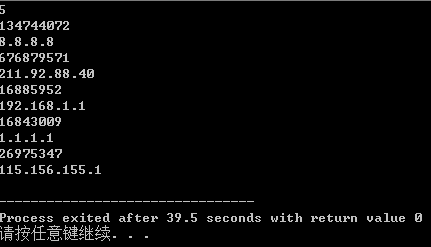


Figure 2 The running results of the test case of programming question 2

It shows that the above operation results are consistent with the theoretical analysis , which verifies the correctness of the program.

## 1.2 Self-set questions

## 1.3 Summary

Although the error points of the previous error correction questions are basic , the most basic place is the place where the basic skills are most tested . Although I saw a lot of mistakes, I didn’t know or how to fix them . I had no choice but to read the book. There were also mistakes like the definition type did not match the input type . I couldn’t see it for a while. The position can only be seen by outputting r again and again to check. All in all, it exposed many of my basic knowledge problems, and urged me to continue reading books and master the basic skills by heart.

Thanks to the C++ training last semester , I quickly completed the second problem , but something went wrong with the third problem. In the extraction of integers, I started to use the loop body as a logic scale, but found that the loop terminated inexplicably early when testing with the set of numbers "555 2 5" ( still looking for the reason ) . But this is also a good thing . It forced me to use bit operations that I was not familiar with at the time to make logic rulers . Thanks to this mistake , I now have a more proficient use of logic rulers and a deeper understanding.

this experiment, I tried different types of development environments, and practiced some simple small programs by myself. During the process of developing the program, I found that some small problems often destroy the normal operation of the entire program, and it is not easy Found, so I think that in the process of our program development, we must ensure the format of the entire source code, and be very careful. At present, I can use simple function calls , distinguish the types of data, and master the use of several loop statements , but they are not proficient enough . I will continue to improve myself in the future practice process and become a basic professional. Literate program developers.

# 2 Process control experiment

## 2.1 Required questions

⑴ Program error correction: 1 question

The following is the source program for calculating s=n! There are several grammatical and logical errors in this source program. It is required to debug and modify this sample program on the computer so that it can correctly complete the specified task. For example, 8! =40320.

1 #include <stdio.h>

2 void main(void)

3 {

4 int i,n,s=1;

5 printf("Please enter n:");

6 scanf("%d",n);

7 for(i=1,i<=n,i++)

8 s=s\*i;

9 printf("%d! = %d",n,s);

10 }

**answer:**

( 1 ) Error modification:

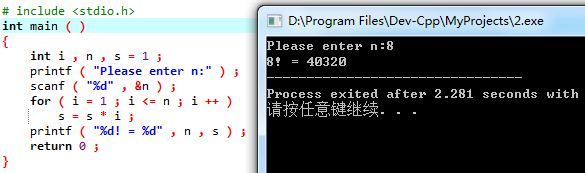
1) The value of n in line 6 has not been modified , and the correct form is:

scanf("%d",&n);

2) The loop condition delimiter on line 7 is wrong , and the correct form is:

for(i=1 ; i<=n;i++)

( 2 ) Running result after error modification:



(2) Program modification and replacement: 2 questions

1. Modify question 1.2.1 and replace the for statement with while and do-while statements respectively.

**answer :**

The replaced program looks like this :

//while statement

# include <stdio.h>

int main ( )

{

int i = 1 , n , s = 1 ;

printf ( "Please enter n:" ) ;

scanf ( "%d" , &n ) ;

while ( i <= n )

{

s \*= i ;

i++;

}

printf ( "%d! = %d" , n , s ) ;

return 0 ;

}

// do-while statement

# include <stdio.h>

int main ( )

{

int i = 1 , n , s = 1 ;

printf ( "Please enter n:" ) ;

scanf ( "%d" , &n ) ;

do

{

s \*= i ;

i++;

}

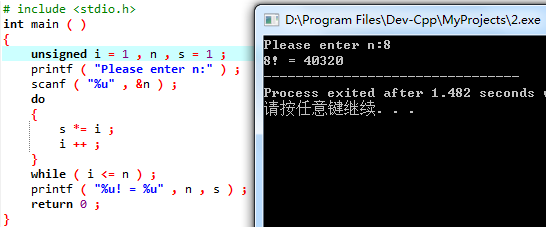
while ( i <= n ) ;

printf ( "%d! = %d" , n , s ) ;

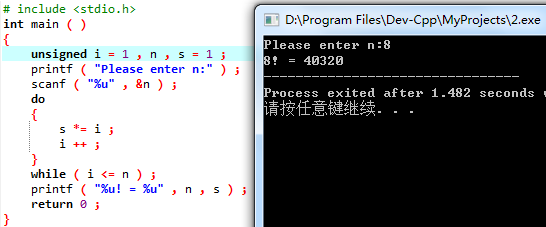
return 0 ;

}

while under:



Under do-while :



2. Modify question 1, change the input to "integer S", and change the output to "the smallest integer n satisfying n!≥S". For example, input the integer 40310, and the output result is n=8.

**answer :**

# include <stdio.h>

int main ( )

{

unsigned i = 1 , n , s = 1 , S ;

printf ( "Please enter S:" ) ;

scanf ( "%u" , &S ) ;

while ( s < S )

{

i++;

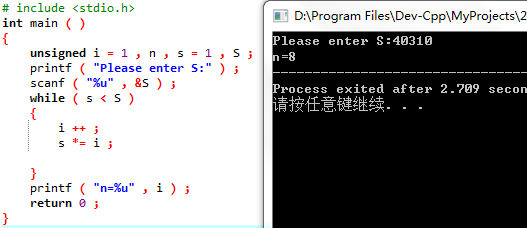
s \*= i ;

}

printf ( "n=%u" , i ) ;

return 0 ;

}



(3) Programming: 4 questions

1. Personal income tax ( experimental questions )

**answer:**

1) Problem-solving ideas:

1. Enter salary money;

2. Judge the value of money , set the condition of if according to the interval given in the title and put money into if;

2.1 money< 1000 , do not pay tax;

2.2 money > 1000 && money <= 2000, the part exceeding 1000 will be charged %5;

2.3 For money > 2000 && money <= 3000, the fixed payment is 50, and the part exceeding 2000 is paid %10;

2.4 money > 3000 && money <= 4000, the fixed payment is 150, and the part exceeding 3000 is paid %15;

2.5 For money > 4000 && money <= 5000, the fixed payment is 300, and the part exceeding 4 000 is paid %20;

2.6 For money > 5000, the fixed payment is 500, and the part exceeding 5000 is paid %25;

3. end

2 ) Program List

int main()

{

double money;

scanf ( "%lf" , &money ) ;

while ( money != 0 )

{

if ( money <= 1000 ) // no tax

printf ( "%f\n" , 0 ) ;

else if ( money > 1000 && money <= 2000 )

//1000<x<=2000 , collect %5 tax

printf ( "%f\n" , ( money - 1000 ) \* .05 ) ;

else if ( money > 2000 && money <= 3000 )

//2000<x<=3000 , collect %10 tax

printf ( "%f\n" , 1000 \* .05 + ( money - 2000 ) \* .1 ) ;

else if ( money > 3000 && money <= 4000 )

//3000<x<=4000 , collect %15 tax

printf ( "%f\n" , 1000 \* .15 + ( money - 3000 ) \* .15 ) ;

else if ( money > 4000 && money <= 5000 )

//4000<x<=5000 , collect %20 tax

printf ( "%f\n" , 1000 \* .3 + ( money - 4000 ) \* .2 ) ;

else //x>5000, collect %25 tax

printf ( "%f\n" , 1000 \* .5 + ( money - 5000 ) \* .25 ) ;

scanf ( "%lf" , &money ) ;

}

}

3 ) Test

(a) Test data:

Table 2-1 Test data of programming question 1

|  |  |  |  |
| --- | --- | --- | --- |
| test  Example | program input | theoretical results | operation result |
| no |
| use case 1 | 500 | 0.000000 | 0.000000 |
| use case 2 | 1500 | 25.000000 | 25.000000 |
| use case 3 | 2500 | 100.000000 | 100.000000 |
| use case 4 | 3500 | 225.000000 | 225.000000 |
| use case 5 | 4500 | 400.000000 | 400.000000 |
| use case 6 | 5500 | 625.000000 | 625.000000 |

It shows that the above operation results are consistent with the theoretical analysis , which verifies the correctness of the program.

2. Space processing ( experimental questions )

**answer:**

1) Problem-solving ideas:

1. Preparatory work : i, n control the number of loop groups, the length of the character array is 200: large and small; getchar() separates the number of character groups and the character text ;

2. Space processing:

2.1 First, the flag is set to 0 , and the current character can be output at this time , and the subsequent change of flag 1/0 determines whether the current character is output , 1: yes /0: no;

2.2 Use count to save the total number of output characters as a check ( especially when there is a string of spaces at the end of the string ) ;

2.3 When the current character is not a space, the flag takes the default value , and at the same time, when the pointer moves from a space to a non-space, the flag is changed back to the default value ;

2.4 When the current character is the first character space or the first space between characters , since the flag is still 0 at this time, this space can be output normally , but the flag will be changed to 1 after the output work;

2.5 When the flag is 1 , if the current character is not a space, the flag will be modified, otherwise it will not be output ;

3. When the pointer points to a null character, one round of output work ends , and the next round of input is performed;

4. end

2 ) Program List

int main()

{

unsigned i , j , n , count , flag ;

char sen[200];

scanf ( "%u" , &n ) ;

getchar();

for ( i = 0 ; i < n ; i ++ )

{

flag = count = 0 ; // judging by flag whether to output characters in the current state

fgets( sen , 200 , stdin ) ;

for ( j = 0 ; sen[j] != '\n' ; j ++ )

{

if ( sen[j] != ' ' ) flag = 0 ;

//Change the flag when the character is cut into by a space, and continue to output

if ( flag ) continue ;

//Do not output in a series of spaces

if ( ! ( flag ) )

{

putchar ( sen[j] ) ; // Normal output when the current character is a non-space character

count ++ ; //count is used to assist in checking output

if ( sen[j] == ' ' )

flag = 1;

// Change the flag when a non-empty character is cut into a space, and determine whether the flag is changed by the judgment result of the subsequent character

}

}

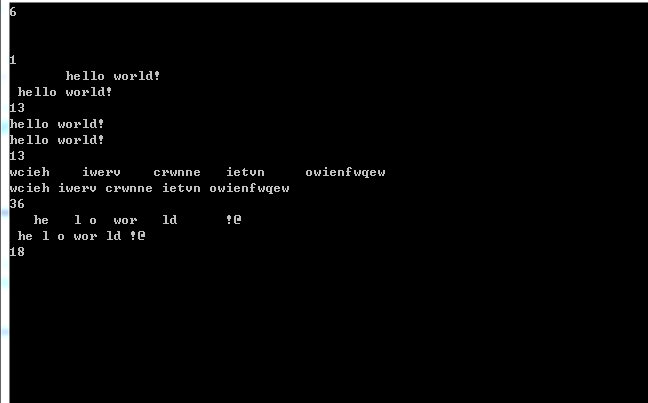
printf ( "\n%d" , count ) ;

putchar ( '\n' ) ;

}

}

3 ) Screenshots of test data and corresponding running results



It shows that the above operation results are consistent with the theoretical analysis , which verifies the correctness of the program.

3. Print the Yang Hui triangle as follows.

1 /\*line 0\*/

1 1 /\*line 1\*/

1 2 1 /\*line 2\*/

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

Each data value can be calculated by combination (representing the value of row i, column j), and the calculation of is as follows:

 (i=0,1,2, ...)

 (j=0,1,2,3, ...,i)

In order to print out the pyramid effect in this program, pay attention to the number of spaces. There are 3 spaces between one-digit numbers, 2 spaces between two-digit numbers, and only one space between 3-digit numbers. Pay attention to the distinction during programming.

**answer:**

1) Problem-solving ideas:

1. Since the number of rows is less than 13, Yang Hui's triangle can satisfy the title format, so the two-dimensional array is defined as a square matrix of 13 \*13 ;

2. Generate Yang Hui triangle:

2.1 Set the first column of the matrix and the elements on the main diagonal to 1;

2.2 From the third line, other numbers of Yang Hui's triangle are generated , that is, the sum of the upper and upper left elements at this position in the square matrix ;

3. Output Yang Hui triangle on demand:

3.1 Determine the number of output lines ;

3.2 The number of output data in each column is equal to the number of rows in the current row ;

4. end

2 ) Source program list

# include <stdio.h>

int main()

{

short i , j , n , blank , tri[13][13] = {0} ;

//When the number of rows in the array is within 13, it is possible to output in the format required by the title

for ( i = 0 ; i < 13 ; i ++ )

{

tri[i][0] = tri[i][i] = 1;

//Initialize the two waists of Yang Hui's triangle, that is, the leftmost column and the main diagonal of the matrix

}

for ( i = 2 ; i < 13 ; i ++ )

{

for ( j = 1 ; j < i ; j ++ )

//Generate the entire Yanghui triangle based on the two waists of the Yanghui triangle for easy access

{

tri[i][j] = tri[i-1][j-1] + tri[i-1][j] ; // Yang Hui triangle recursion formula

}

}

scanf ( "%hd" , &n ) ;

while ( n != 0 )

{

for ( i = 0 ; i < n ; i ++ )

{

blank = 2 \* ( n - 1 - i ) ;

/\*Observing the output format, it can be found that the data output line width is 4, and the data in the Yang Hui triangle is aligned every other line, so the line indentation difference is 2\*/

for ( j = 1 ; j <= blank ; j ++ ) // Adjust space to output isosceles triangle

putchar ( ' ' ) ;

for ( j = 0 ; j <= i ; j ++ ) // output the first n rows of Yang Hui's triangle in sequence

printf ( "%-4hd" , tri[i][j] );

putchar ( '\n' ) ;

}

putchar ( '\n' ) ;

scanf ( "%hd" , &n ) ;

}

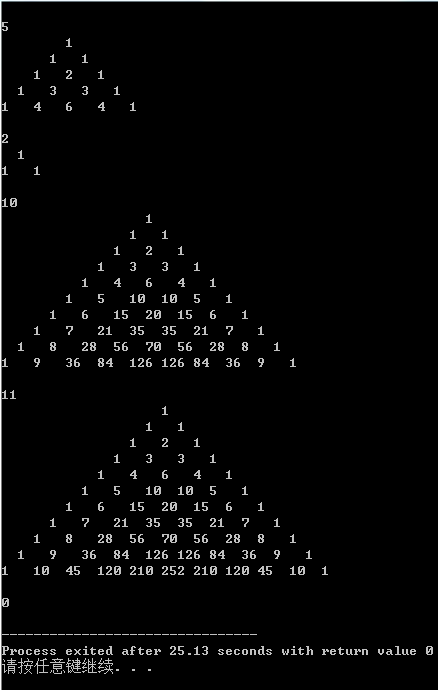
}

3 ) Test

( a ) Test Data:

5 , 2, 10, 11, 0

( b ) Screenshot of the running results corresponding to the test data



It shows that the above operation results are consistent with the theoretical analysis , which verifies the correctness of the program.

4. Write a program to reverse any positive integer input by the user, for example, input 1234 and output 4321.

**answer:**

1) Problem-solving ideas:

1. The title needs to judge each input data, so it is necessary to add an input before the while statement

2. Extract and output digits .

2.1 First n % 10 , extract the single digit and output it ;

2.2 n /= 10 ensures that n%10 extracts each digit of the original data from small to large ;

When n=0, the reverse sequence output work ends, and the next input is performed ;

3. The program ends when the input is 0 ;

2 ) Program List

# include <stdio.h>

# include <stdlib.h>

int main()

{

unsigned long long n ;

//Use unsigned long long to hold the input data whose absolute value is as large as possible

scanf ( "%llu" , &n ) ;

while ( n != 0 )

{

while ( n != 0 )

{

printf ( "%d" , n % 10 ) ; n /= 10 ; // Extract single digit and output

}

putchar ( '\n' ) ; scanf ( "%llu" , &n ) ;

}

}

3 ) Test

(a) Test data:

the maximum data that can be stored normally is 2^64 -1

Table 2-2 Test data of programming question 4

|  |  |  |  |
| --- | --- | --- | --- |
| test  Example | program input | theoretical results | operation result |
| no |
| use case 1 | 2^64-1 | 61,615,590,737,044,764,481 | 61,615,590,737,044,764,481 |
| use case 2 | 2^64 | r return 0 | r return 0 |
| use case 3 | 2^64+1 | 1 (overfloat) | 1 |

It shows that the above operation results are consistent with the theoretical analysis , which verifies the correctness of the program.

## 2.2 Optional questions

Programming: 1 question

Find approximate roots of equations ( experimental questions , optional questions)

**answer:**

1) Problem-solving ideas:

1. Set the original function and derivative function according to the topic ; since the approximate root of the equation obtained by the Newton iteration method has nothing to do with the initial value of the iteration , x1 can be assigned arbitrarily ;

2. Obviously, the termination condition of the loop is that the difference between the two numbers iterated is less than the given error 1 e-6 , use x1 and the given formula to calculate the value of x2 , if the difference between the two numbers does not meet the termination condition, then assign x2 to x1, Start the next cycle ;

3. The loop is terminated, output x2, and end .

2 ) Program List

# include <stdio.h>

int main ( )

{

double x1 = 0.0 , x2 , ori , dif ;

x2 = x1 + 1 ;

while ( fabs ( x1 - x2 ) > 1e-6 )

{

x1 = x2;

ori = 3 \* pow ( x1 , 3 ) - 4\* x1 \* x1 - 5 \* x1 + 13 ;

dif = 9 \* x1 \* x1 - 8 \* x1 - 5 ;

x2 = x1 - ori / dif ;

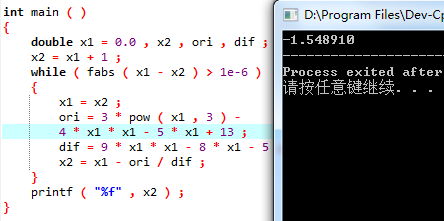
}

printf ( "%f" , x2 ) ;

}

3 ) Test

Screenshot of the running result :



It shows that the above operation results are consistent with the theoretical analysis , which verifies the correctness of the program.

## 2.3 Self-made questions

**(1)** Self-made experiment topic: One rooster is worth five coins , one hen mother is worth three coins, three chicks are worth one coin. Buy a hundred chickens for a hundred dollars , ask the rooster , how much are the mothers and chicks?

**(2)** Purpose of the experiment: Through the design of the experimental program, understand the law that the calculation time increases with the power of the number of loop layers in multiple loops , and understand that the running time can be effectively shortened through algorithm optimization. .

**(3)** Experimental procedure:

#include < stdio.h>

int main()

//Original algorithm - triple loop

#include <stdio.h>

{

short i , j , k ;

printf ( " rooster hen chick\n" );

for ( i = 0 ; i <= 20 ; i ++ )

for ( j = 0 ; j <= 33 ; j ++ )

for ( k = 0 ; k <= 300 ; k ++ )

if ( ( i + j + k == 100 ) && ( 5 \* i + 3 \* j + k / 3 == 100 ) && ( k % 3 == 0 ) )

printf ( "%6hd%6hd%6hd\n" , i , j , k ) ;

return 0 ;

}

// Improved algorithm - double cycle

int main ( )

{

short i , j , k ;

printf ( "rooster hen chick\n" );

for ( i = 0 ; i <= 20 ; i ++ )

{

for ( j = 0 ; j <= 33 ; j ++ )

{

k = 100 - i - j;

if ( ( 5 \* i + 3 \* j + k / 3 == 100 ) && ( k % 3 == 0 ) )

printf ( "%6hd%6hd%6hd\n" , i , j , k ) ;

}

}

return 0 ;

}

**(4)** Experimental use cases:

Original algorithm running time :





Improved algorithm running time :





**(5)** Experimental conclusion: Theoretically speaking , the loop body in the original algorithm will execute 20 \*33\*300=198000 times , and the improved algorithm only needs to execute 20 \*33=660 times , but the shortening of the running time is not obvious . This is because the program is small, the optimization of the algorithm is not fully reflected , and the double loop is a few more calculation steps than the triple loop. In larger and more complex programs , the reduction in program execution time will be more pronounced.

## 2.4 Summary

In this experiment, I became more familiar with Dev's operation. Knowing some small mistakes that are easy to make when writing programs, you have to be careful everywhere in the future programming road. Specifically, I found that float variables cannot be input in the scanf function, otherwise errors will occur. In addition, I have become more proficient in the function calling process, but overall, the speed of writing programs is still quite slow, and it needs to be improved in the future .

The training of source program error correction and modification and replacement and the knowledge reserve of C ++ in the last semester made me familiar with the usage methods and situations of the three loop bodies , but my current application is still relatively rigid, and the construction of the three statements of the for loop It is still at the simple level of a single statement , and you should practice more later to master the flexible use of for statements such as multiple initial conditions, multiple self-increment steps, complex and diverse judgment conditions , and empty statements . Regarding the while statement and the do-while statement , I think while is closer to our language logic. The situation of " cutting first and then playing " like do -while rarely occurs in daily life, but the so-called existence is reasonable. Many practical scenarios of do-while need Through future programming a little bit to explore.

Space handling can be said to be the question that has stuck with me for the longest time . First of all, because I was frightened by the topic to a certain extent at the beginning , I had a certain degree of fear, and secondly, because I didn't have a clear and complete algorithm system when I conceived it, I wrote it directly when I thought of it, which later caused me to I can't even figure out how my program works. Therefore, sharpening a knife is not a mistake in chopping firewood. Before typing the code, you must first outline a clear algorithm structure in your mind, so that you will not feel like " I don't quite understand what I am writing " . At the same time, through this question, I am also more skilled in using the method of using print f to check the running process of the program, but this is only a last resort when I am not familiar with single-step debugging . I should master this method as soon as possible to check the program. more efficient.

Yang Hui's triangle is another pitfall for me . I didn't encounter any major problems in the algorithm , but because I didn't observe the sample output format carefully, I always thought that I could use the left output processing and ignored the difference in the space between the number and the number under the two output methods. Sometimes the requirements that are not written in the text are reflected in the sample output. You must observe them carefully in the future, and you must not skip over carelessly and miss many details.

The problem of number inversion can actually be solved by using the array knowledge later , which solves the problem of limited value range of data types; it can also be solved by using linked list knowledge, which solves the problem that the array length must be determined in advance when processing arrays . However, when the type of data to be processed is relatively simple and the absolute value is small, the latter two methods are a bit overkill , so bitwise extraction is actually sufficient here .

# 3 Function and program structure experiment

## 3.1 Required questions

⑴ Program error correction: 1 question

The following is the source program for calculating s=1!+2!+3!+…+n! There are several syntax and logic errors in this source program. It is required to debug and modify this sample program on the computer so that it can correctly complete the specified task.

1 #include "stdio.h"

2 void main(void)

3 {

4 int k;

5 for(k=1;k<6;k++)

6 printf("k=%d\tthe sum is %ld\n",k,sum\_fac(k));

7 }

8 long sum\_fac(int n)

9 {

10 long s=0;

11 int i;

12 long fac;

13 for(i=1;i<=n;i++)

14 fac\*=i;

15 s+=fac;

16 return s;

17 }

**answer:**

( 1 ) Error modification:

1) The function starting from line 8 is not declared , and the correct form is:

(insert between line 1 and line 2) long sum\_fac ( int n ) ;

2) The fac in line 12 is not initialized , the correct form is:

long fac = 1;

3 ) The statements on lines 14 and 15 are not loaded into a loop section, and the correct form is:

for(i=1;i<=n;i++)

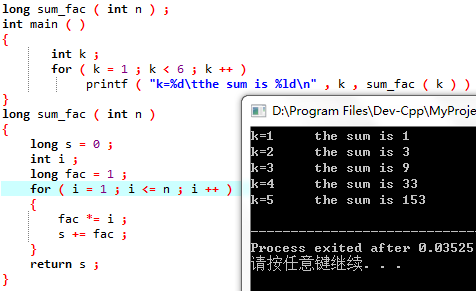
{

fac\*=i;

s+=fac;

}

( 2 ) Running result after error modification:



(2) Program modification and replacement: 2 questions

1. Modify the sum\_fac function in question 1 to minimize the amount of calculation.

**answer :**

#include <stdio.h>

long sum\_fac(int n)

{

static long s = 0 , fac = 1 ;

fac \*= n ;

s += fac ;

return s;

}

int main ( void )

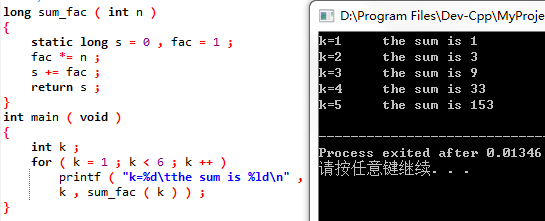
{

int k ;

for ( k = 1 ; k < 6 ; k ++ )

printf ( "k=%d\tthe sum is %ld\n" , k , sum\_fac ( k ) ) ;

}



2. Modify the sum\_fac function in question 1 to calculate .

**answer :**

long myfac ( int n )

{

if ( n == 0 || n == 1 )

return 1 ;

else

return n \* myfac ( n - 1 ) ;

}

double sum\_fac ( int n )

{

if ( n == 0 || n == 1 )

return 1 ;

else

return 1 / ( double )( n \* myfac ( n - 1 ) ) + sum\_fac ( n - 1 ) ;

}

int main ( )

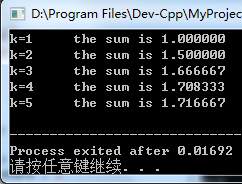
{

int k ;

for ( k = 1 ; k < 6 ; k ++ )

printf ( "k=%d\tthe sum is %f\n" , k , sum\_fac ( k ) ) ;

}



(3) Tracking and debugging: 4 questions

The program to calculate the sum of the first n terms of the fabonacci sequence is as follows:

Among them, long sum=0, \*p=&sum; declare p as a long integer pointer and use &sum to take out the address of sum to initialize p. \*p refers to the variable pointed to by p (\*p is sum).

void main(void)

{

int i,k;

long sum=0,\*p=&sum;

scanf("%d",&k);

for(i=1;i<=k;i++){

sum+=fabonacci(i);

printf("i=%d\tthe sum is %ld\n",i,\*p);

}

}

long fabonacci(int n)

{

if(n==1 || n==2)

return 1;

else

return fabonacci(n-1)+fabonacci(n-2);

}

Step through the program and observe the p,i,sum,n values.

1. Just finished executing the scanf("%d",&k); statement, what are the values of p and i?

p: 0x16 (randomly assigned address); i: 2;

1. On which statement does the light bar stay after returning from a fabonacci function?

printf("i=%d\tthe sum is %ld\n",i,\*p);

1. Enter the fabonacci function, what does the watch window display?

Function arguments: n — 2; Locals: (none);

(4) When i=3, how does the value of n change from calling the fabonacci function to returning?

n=3 => n=2 => n=1 ;

⑷Programming : 2 questions

1. Programming allows the user to input two integers, calculate the greatest common divisor of the two numbers and output it (it is required to use a recursive function to find the greatest common divisor). At the same time, execute the program in a single step and observe the recursive process.

**answer:**

1) Problem-solving ideas:

1. Determine whether the input a and b meet the conditions in advance , and calculate the common divisor only when the cycle condition is true ;

2. Find the greatest common divisor by subtraction;

2.1 Recursive termination condition : the larger of the two numbers is divisible by the smaller one , and the smaller value is returned when the condition is satisfied , which is the greatest common divisor of the original two numbers ;

2.2 When the condition is not met, return the absolute value of the difference between b and the two numbers . This is considering that return maxdiv ( b , abs ( a - b ) ) is an infinite loop when a>b is returned if a is returned ;

3. Continue to input a and b to make the next judgment of the cycle condition;

4. Jump out of the loop when a is 0 , and the program ends ;

2 ) Source program list

int maxdiv ( int a , int b )

{

if ( a % b != 0 )

return maxdiv ( b , abs ( a - b ) ) ;

//From the third layer of recursion, output the smaller number of a and b and the absolute value of the difference between the two numbers stably

return b;

}

int main()

{

int a , b ;

scanf ( "%d %d" , &a , &b ) ;

while ( a != 0 )

{

printf ( "%d\n" , maxdiv ( a , b ) ) ;

scanf ( "%d %d" , &a , &b ) ;

}

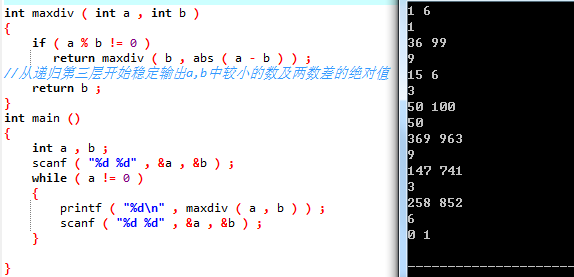
}

3 ) Test

( a ) Test Data:

1&6 36&99 15&6 50&100 369&963 147&741 258&852 0&2

( b ) Screenshot of the running results corresponding to the test data



2. Programming to verify Goldbach's conjecture: an even number greater than or equal to 4 is the sum of two prime numbers.

Write a program to prove this guess holds for even numbers between the symbolic constants BEGIN and END. For example, if BEGIN is 10 and END is 20, the output of the program should be:

GOLDBACH'S CONJECTURE:

Every even number n>=4 is the sum of two primes.

10=3+7

12=5+7

...

20=3+17

**answer:**

1) Problem-solving ideas:

1. Output judgment interval START and END;

2. To judge START and END , if ST ART is odd , then +1 ; if END is even, then -1 ;

3. Perform operator judgment for each even number between START and END , and output the formula if the operators are all prime numbers;

4. Enter the operator judgment function , flag is used as a flag variable, if any integer from 2 to its square root takes a remainder of 0, then it is not a prime number , and enters the next cycle.

5. Return the (non-)flag value ;

6. end;

2 ) Program List

char prime ( long n )

{

char flag = 1 ; long i = 2 ;

for ( ; i <= sqrt ( n ) ; i ++ )

if ( n % i == 0 ) // If it is not a prime number, flag=0, correspondingly the if statement in main is judged as false

return ! ( flag ) ;

return flag;

}

int main()

{

long a , b , i , j , BEGIN , END ;

scanf( "%ld %ld" , &BEGIN , &END ) ;

while ( BEGIN != 0 )

{

if ( BEGIN < 6 || END < 6 || END < BEGIN ) // Illegal input intervention

{ printf ( "error!\n" ) ; }

else

{

if ( BEGIN % 2 )

a = BEGIN + 1 ;

else

a = BEGIN ; // set the loop initial value (even number)

if ( END % 2 )

b = END - 1;

else

b = END ; // set the end value of the loop (even number)

for ( i = a ; i <= b ; i += 2 )

for ( j = 2 ; j <= i / 2 ; j ++ )

if ( prime ( j ) && prime ( i - j ) ) // judging whether both operators are prime numbers

{

printf ( "%ld=%ld+%ld\n" , i , j , i - j ) ;

break;

}

}

putchar ( '\n' ) ;

scanf( "%ld %ld" , &BEGIN , &END ) ;

}

}

3 ) Test

the corresponding test cases are shown in Figure 1-2.

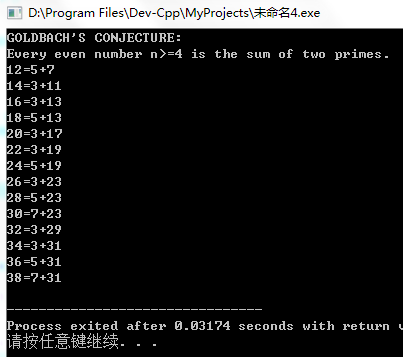


Figure 3-2 The running results of the test case of programming question 2

It shows that the above operation results are consistent with the theoretical analysis , which verifies the correctness of the program.

## 3.2 Optional questions

Programming: 1 question

Let file1.c be as follows:

#include <stdlib.h>

#include "HEADER.h"

int x,y; /\* Definition of external variables \*/

char ch; /\* Definition of external variables \*/

int main(void)

{

x=10;

y=20;

ch = getchar();

printf("in file1 x=%d,y=%d,ch is %c\n",

x,y,ch);

func1();

return 0 ;

}

file2.c is as follows:

#include "HEADER.h"

void func1(void)

{

x++;

y++;

ch++;

printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);

}

HEADER.h is as follows:

#ifndef HEADER\_H\_INCLUDED

#define HEADER\_H\_INCLUDED

#include <stdio.h>

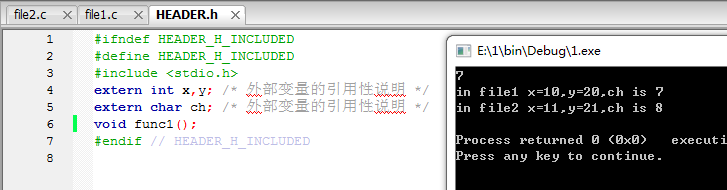
extern int x,y; /\* referential description of external variables \*/

extern char ch; /\* reference description of external variable \*/

void func1();

#endif // HEADER\_H\_INCLUDED

Try TCC for multi-file compilation and linking. Then run the generated executable file under DOS environment.



## 3.3 Self-made questions

Self-set experiment topic: define three functions , find the sum of two integers, three integers and three floating-point numbers respectively

**(2)** Purpose of the experiment: Get familiar with the characteristics and principles of function overloading

**(3)** Experimental procedure:

#include < stdio.h>

int sum ( int a , int b ) //int type, double parameter

{

return a + b;

}

int sum ( int a , int b , int c ) //int type, three parameters

{

return a + b + c ;

}

double sum ( double a , double b , double c ) //double type, three parameters

{

return a + b + c ;

}

int main()

{

int a , b , c ;

double x , y , z ;

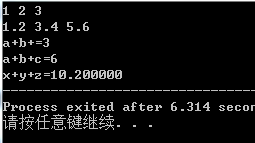
scanf( "%d %d %d" , &a , &b , &c ) ;

scanf ( "%lf %lf %lf" , &x , &y , &z ) ;

printf ( "a+b+=%d\na+b+c=%d\nx+y+z=%f" , sum ( a , b ) , sum ( a , b , c ) , sum ( x , y , z ) ) ;

}

**(4)** Experimental use cases: a =1;b=2;c=3;x=1.2;y=3.4;z=5.6;



**(5)** Experimental conclusion: Function overloading can reduce the number of function names and improve their readability. However, it should be noted that to achieve overloading, at least the number or type of formal parameters must be different to avoid ambiguity in function calls; the compiler will generate call matches based on different parameter types and numbers.

## 3.4 Summary

Although recursion has been touched in C++ , it is just a touch. This is the first time that I have used recursion frequently on a large scale. Although I have to get stuck on it for a while every time, I passed it smoothly afterwards. To correctly apply and master recursion , it is important to have a deep understanding of the specific implementation method of the program, the recursive algorithm , the characteristics of passing values, and the order of calculation. At the same time , although the recursive algorithm has a compact structure, clear structure, strong readability, and concise code, its operating efficiency is low , the calculation time is long, and the storage space occupied is particularly large. Therefore, when using recursion to write programs, pay attention to whether the program itself has computational efficiency requirements and whether the memory can prepare enough space for recursive operations.

I didn't encounter any major obstacles in the programming problem this time . The only thing worth noting is that the recursive value in the recursive search for the greatest common divisor should be a and abs ( a - b ) instead of b and abs ( a - b ) . Although such small mistakes are easy to correct, it is precisely this kind of small bugs that are not easy to find out in team development in the future. So when designing programs in the future , you must think clearly at the beginning , and don't wait until the end to let yourself be confused by a bunch of small mistakes accumulated .

a new technique that has not been touched before . It took a while to get familiar with the operation , and although the debugging was successful, the proficiency is still not good, especially the Step out and Run to cursor that are not required by the title are still unfamiliar . When you go back, you still have to practice more when you program yourself . Single-step debugging can effectively improve the speed of program debugging , but the premise is to have a deep understanding of its use. At the same time, it should be noted that the display style and method of single-step debugging are different for different compilers . For example, Dev -C++ observes The first address of character array will be displayed, but Codeblocks will not. Therefore, you should also master the method of displaying different content (value or address) in the watch window in case of emergency.

When the program body is small and the function relationship is simple, it is a good method to advance the function body ahead of the main function, which can save the declaration of the function and reduce the amount of code to a certain extent. However, when there are many functions and the relationship between mutual calls is complicated, putting the called functions in the front will not only cause inconvenience when reading the program, but also cause confusion or appear when functions call each other. The case where the function is not declared. Therefore, in this case, the called function should be placed after the main function and the declaration should be added in front of it. At the same time, the order of declaration and the writing order of the called function should be consistent.

Although it is the first time I have come into contact with multi-file compilation , it is still relatively simple to get started, but the program size is still small at present , and there is no chance of putting this method into practice. To really master it, you still have to find a program to write the block file yourself .

# 4 Compilation preprocessing experiment

## 4.1 Required questions

⑴ Program error correction: 1 question

1. The following is a source program that uses a macro to calculate the square difference and exchange two numbers. There are several grammatical and logical errors in this source program. It is required to debug and modify this sample program on the computer so that it can correctly complete the specified task.

1 #include "stdio.h"

2 #define SUM a+b

3 #define DIF ab

4 #define SWAP(a,b) a=b,b=a

5 void main

6 {

7 int b, t;

8 printf("Input two integers a, b:");

9 scanf("%d,%d", &a,&b);

10 printf("\nSUM=%d\n the difference between square of a and square of b is: %d ",SUM, SUM\*DIF);

11 SWAP(a,b);

12 Printf("\nNow a=%d,b=%d\n",a,b);

13 }

**answer:**

( 1 ) Error modification:

1) The variable definition in line 7 is wrong , the correct form is:

int b, a ;

2) The form of the no-argument macro definition in line 2 is wrong , and the correct form is:

#define SUM ( a+b )

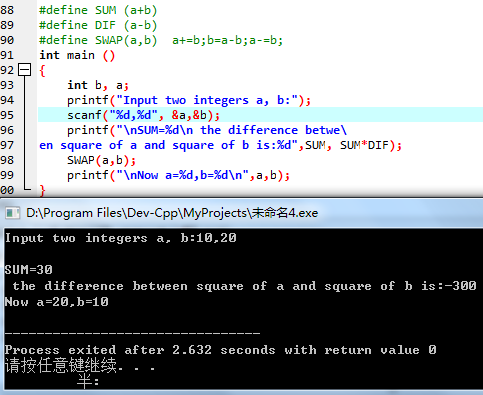
3) The form of the no-argument macro definition in line 3 is wrong , and the correct form is:

#define DIF ( ab )

2) The algorithm of the macro definition with parameters in line 4 is wrong , and the correct form is:

#define SWAP(a,b) a+=b;b=ab;a-=b;

( 2 ) Running result after error modification:



(2) Program modification and replacement: 2 questions

The following is a program that uses functions to find the largest number among three numbers and calculate the sum of two numbers. There are some grammatical and logical errors in this source program.

Requirements: 1) Debug and modify this example program so that it can complete the specified tasks correctly;

2) Replace the function max with a macro with parameters to realize the function of finding the maximum number.

void main(void)

{

int a, b, c;

float d, e;

printf("Enter three integers:");

scanf("%d,%d,%d",&a,&b,&c);

printf("\n the maximum of them is %d\n",max(a,b,c));

printf("Enter two floating point numbers:");

scanf("%f,%f",&d,&e);

printf("\n the sum of them is %f\n",sum(d,e));

}

int max(int x, int y, int z)

{

int t;

if (x>y)

t=x;

else

t=y;

if (t<z)

t=z;

return t;

}

float sum(float x, float y)

{

return x+y;

}

**answer :**

The replaced program looks like this :

#include <stdio.h>

# define max( a , b , c ) a > b ? ( a > c ? a : c ) : ( b > c ? b : c )

float sum ( float x , float y )

{

return x + y ;

}

int main(void)

{

int a , b , c ;

float d , e ;

printf ( "Enter three integers:" ) ;

scanf ( "%d,%d,%d" , &a , &b , &c ) ;

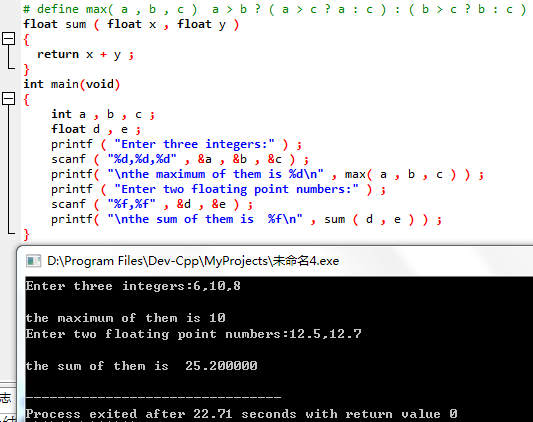
printf( "\nthe maximum of them is %d\n" , max( a , b , c ) ) ;

printf ( "Enter two floating point numbers:" ) ;

scanf ( "%f,%f" , &d , &e ) ;

printf( "\nthe sum of them is %f\n" , sum ( d , e ) ) ;

}



(3) Tracking and debugging: 3 questions

The following program uses R to calculate the area s of a circle, and the integer part of the area s.

#define R

void main(void)

{

float r, s;

int s\_integer=0;

printf("input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

printf("area of round is: %f\n", s);

s\_integer = integer\_fraction(s);

printf("the integer fraction of area is %d\n", s\_integer);

assert((s-s\_integer)<1.0);

#endif

}

int integer\_fraction(float x)

{

int i=x;

return i;

}

1) Modify the program to make the program compile and run;

# include <assert.h> // add

# include <stdio.h>

#define R

int integer\_fraction ( float x )

{

int i = x;

return i;

}

int main ( )

{

float r , s ;

int s\_integer = 0;

printf ( "input a number: " ) ;

scanf ( "%f" , &r ) ;

#ifdef R

s = 3.14159 \* r \* r;

printf ( "area of round is: %f\n" , s ) ;

s\_integer = integer\_fraction ( s ) ;

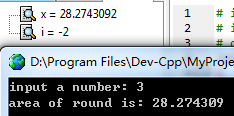
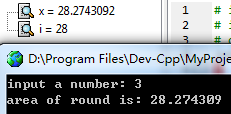
printf ( "the integer fraction of area is %d\n" , s\_integer ) ;

assert ( ( s - s\_integer ) < 1.0 ) ;

#endif

}

2) Single step execution. What is the value of x in the watch window when entering the function decimal\_fraction ? When returning to main , what is the value of i in the watch window?

enter : return :

3) Eliminate errors, so that the program can correctly output the integer part of the area s value, and will not output the error message assertion failed. (see 1 ) in procedure )

⑷Programming : 2 questions

1. The area of a triangle is , where a, b, and c are the three sides of the triangle, define two macros with parameters, one is used to find s, and the other is used to find area. Write a program that uses a macro with parameters to calculate the area of a triangle.

**answer:**

1) Problem-solving ideas:

1. Input the lengths of the sides of the triangle a , b , c

2. Realize the loop condition : the input is not the end of the file

3. Realization of macro definition of half perimeter and area

2.1 The semi-perimeter is defined by the macro with parameters. The parameter is the length of the three sides of the triangle, and its value should be received by variables. Otherwise , the macro definition of the semi-perimeter needs to be included in the macro definition of the area, which will increase the number of calculations and the code is too long when defining the macro.

2.2 Macro definition area with parameters , the parameters are the length of the three sides and the received semi-perimeter , the value can be directly output

4. Input and judge the next cycle

5. end

2 ) Source program list

# include <stdio.h>

#define S( a , b , c ) ( a + b + c ) / 2

//Define S as the semi-perimeter of a triangle with a parameter macro

# define AREA( a , b , c , s ) sqrt ( s \* ( s - a ) \* ( s - b ) \* ( s - c ) )

//Define AREA as the triangle area obtained by Helen's formula with parameter macro

int main()

{

double a , b , c , s ;

while ( scanf ( "%lf %lf %lf" , &a , &b , &c) != EOF )

//The double type is recorded as %lf in the input format string

{

s = S( a , b , c ) ;

// Use a variable to catch the semicircumference obtained by the macro definition, and use it as a parameter of the area macro definition

printf ( "%d %f\n" , ( int ) s , AREA( a , b , c , s ) ) ;

//According to the output requirements, the double type s is forced to be converted, and the value of AREA can be directly output in the format

}

}

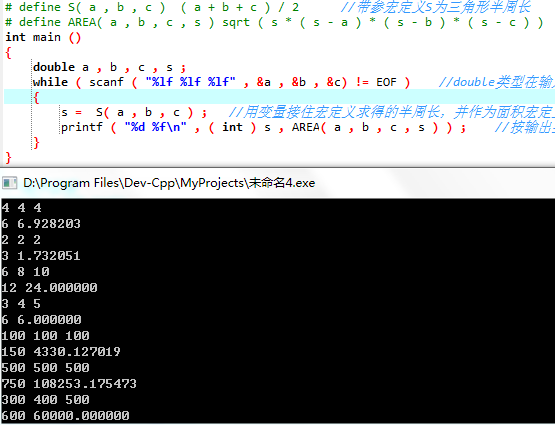
3 ) Test

( a ) Test Data:

4&4&4 2&2&2 6&8&10 3&4&5

100&100&100 500&500&500 300&400&500

( b ) Screenshot of the running results corresponding to the test data



2. Use the conditional compilation method to write the program. Input a line of telegram text, you can choose two outputs: one is the original text output; the other is to change the case of letters (such as lowercase 'a' into uppercase 'A', uppercase 'D' into lowercase 'd'), other The characters are unchanged. Use the #define command to control whether to change the case of letters. For example, #define CHANGE 1 will output the converted text, if #define CHANGE 0, the original text will be output.

**answer:**

1) Problem-solving ideas:

1. Enter n to determine the number of test string groups ;

2. fgets for string input ;

2. Conditional compilation format .

2.1 There is no parameter macro definition before the main function to set the CHANGE value ;

2.2 The if -else structure is replaced by conditional compilation . Note that the constant expression used as the judgment condition should have a value determined during precompilation

3. Conditional compilation structure ;

3.1 CHANGE==0 => Original text output ;

3.2 CHANGE==1 => convert the case of the letter while scanning the string ( if the current character is a letter) , then output

4. Enter the next cycle

5. end

2 ) Program List

#include <stdio.h>

#define CHANGE 0

int main()

{

unsigned short i , j , n ;

char a[500] , c ;

scanf ( "%hu ", &n ) ;

for ( i = 1 ; i <= n ; i ++ )

{

fgets( a , 500 , stdin ) ;

c = a[0];

#if ( ! CHANGE ) //CHANGE==0

printf ( "%s" , a ) ;

#else //CHANGE==1

{

for ( j = 0 ; a[j] != 0 ; j ++ )

// Filter each element of the array, if it is a letter, flip the case

if ( a[j] >= 'a' && a[j] <= 'z' )

a[j] = a[j] -'a' + 'A' ; // lowercase to uppercase

else if ( a[j] >= 'A' && a[j] <='Z' )

a[j] = a[j] -'A' + 'a' ; // uppercase to lowercase

printf ( "%s", a ) ; // string format output

}

#endif

}"

}

3 ) Test

( a ) Test Data:

"abcd!@#$efg";

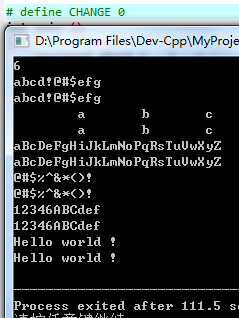
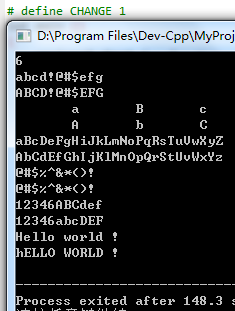
"ab c"; "

"aBcDeFgHiJkLmNoPqRsTuVwXyZ";

"@#$%^&\*()!";

"Hello world!";

( b ) Screenshot of the running results corresponding to the test data



## 4.2 Self-made questions

**(1)** Self-set experiment topic: multi-statement macro definition

**(2)** Purpose of the experiment: Through the design of the experimental program, it is understood that there can be multiple statements in the macro definition , so that the macro definition can be closer to the lightweight function when used.

**(3)** Experimental procedure:

#include "stdio.h"

# define mul( s1 , s2 , s3 , v ) s1 = k \* w ; s2 = k \* h ; s3 = w \* h ; v = w \* k \* h ;

int main()

{

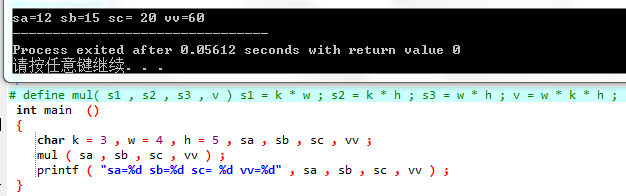
char k = 3 , w = 4 , h = 5 , sa , sb , sc , vv ;

mul ( sa , sb , sc , vv ) ;

printf ( "sa=%d sb=%d sc= %d vv=%d" , sa , sb , sc , vv ) ;

}

**(4)** Experimental use cases:



**(5)** Experimental conclusion: Functions with few statements and simple functions can be replaced by multi-statement macro definitions, thereby reducing the number of parameter transfers and the possibility of errors.

## 4.3 Summary

The chapter on compilation and preprocessing seems relatively simple. I think this is because its knowledge points are relatively simple and easy to write, so the application part is not a problem . But the include part is actually very famous. The simplicity of high-level programming languages lies in the large number of header files and library functions provided by the system , and the implementation of header files and library functions is a part that can be dug deeper. At the same time , compilation preprocessing is also related to the previous knowledge of multi-file compilation and storage types. In the future, we will often write our own header files according to the development environment and requirements . It is also common for variables and functions to be used across files. It is time to fully demonstrate the importance of various compilation preprocessing including include .

When using macro definitions , you must never omit parentheses just because you find it troublesome . The error correction of the source program has already explained that the compiler only performs macro substitution, and does not give priority to the substituted statements . It takes some effort to add parentheses when defining, so that the logic errors it brings will no longer involve time and energy when debugging .

Since I was not familiar with conditional compilation at the beginning , I didn't realize #if when I defined CHANGE / The constant expression after #elif must have a definite value during precompilation and cannot be determined by external input at runtime. At the same time, #endif has no corresponding statement in the conditional structure, so be careful not to omit it when using conditional compilation .

Since the assert assertion is used for the first time, it is inevitable to scan the book again. For this kind of independent knowledge point, it is necessary to use it consciously at ordinary times, so that it can be mastered as soon as possible. Assert assertion combined with conditional compilation and single-step debugging can improve debugging efficiency . The difference is that single-step debugging is easier to check variables, while assert assertion and conditional compilation are more suitable for checking expressions and output problems. The flexible use of the three debugging methods needs to be developed in the future It can only be mastered by repeated repetitions in the programming.

# 5 array experiments

## 5.1 Required questions

⑴ Program error correction: 1 question

The following is the source program used to sort the elements in the array a in ascending order and output them. Analyze the problems existing in the source program, and modify the source program so that it can complete the task correctly.

source program

1 #include <stdio.h>

2 int main(void)

3 {

4 int a[10] = {27, 13, 5, 32, 23, 3, 17, 43, 55, 39};

5 void sort(int [],int);

6 int i;

7 sort(a[0],10);

8 for(i = 0; i < 10; i++)

9 printf("%6d",a[i]);

10 printf("\n");

11 return 0;

12 }

13 void sort(int b[], int n)

14 {

15 int i, j, t;

16 for (i = 0; i < n - 1; i++)

17 for ( j = 0; j < n - i - 1; j++)

18 if(b[j] < b[j+1])

19 t = b[j], b[j] = b[j+1], b[j+1] = t;

20 }

**answer:**

( 1 ) Error modification:

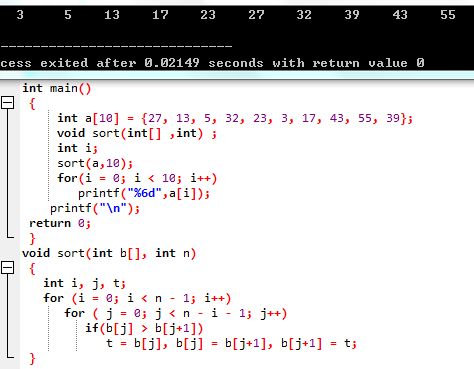
1) The form of the array value transfer in line 7 is wrong , the correct form is:

sort(a,10);

2) The function exchange condition in line 18 is wrong, which leads to the actual sorting in descending order . The correct form is:

if(b[j] > b[j+1])

( 2 ) Running result after error modification:



(2) Program modification and replacement: 2 questions

1. The following source program is used to solve the Cerf problem: M people form a circle, starting from the first person and counting from 1 to N, and counting people out of the circle every time the number is N until they are in the circle Until there is only one person left. Please fill in the appropriate code in the underlined place in the source program to perfect the program.

Source program:

#include <stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M]; /\* Array a stores the number of people in the circle, and array b stores the number of people out of the circle\*/

int i, j, k;

for(i = 0; i < M; i++) /\* number the people in the circle sequentially from 1 to M \*/

a[i] = i + 1;

for(i = M, j = 0; i > 1; i--){

/\* i represents the number of people in the circle, the initial number is M, and the cycle ends when there is 1 person left; j represents the position of the current reporter\*/

for(k = 1; k <= N; k++) /\* report number from 1 to N\*/

if(++j > i - 1) j = 0;/\* After the last person counts, the first person will continue to report, forming a circle\*/

b[Mi] = j? : ; /\* Store the number of the person whose number is N into the array b \*/

if(j)

for(k = --j; k < i; k++) /\* compress the array a, so that the people whose number is N are out of the circle\*/

;

}

for(i = 0;i < M – 1; i++) /\* output the numbers of the circle members in order\*/

printf( " %6d " , b[i]);

printf( " %6d\n " , a[0]); /\* Output the number of the last person in the circle \*/

return 0;

}

**answer :**

#define M 10

#define N 3

int main(void)

{

int a[M], b[M]; // Array a stores the number of people in the circle, array b stores the number of people out of the circle

int i , j , k ;

for(i = 0 ; i < M ; i++ ) // number the people in the circle sequentially 1-M

a[i] = i + 1 ;

for( i = M , j = 0 ; i > 1 ; i-- ) // i represents the number of people in the circle, the initial number is M, and the loop ends when there is 1 person left; j represents the position of the current reporter

{

for(k = 1 ; k <= N ; k++ ) // report from 1 to N

if ( ++j > i - 1 )

j = 0 ; // After the last person counts, the first person will continue to report, forming a circle

b[Mi] = j ? a[j-1] : a[i-1] ; // Store the number of the person whose number is N into the array b

if ( j )

for (k = --j ; k < i ; k++ ) // Compress the array a, so that the people whose number is N are out of the circle

a[k] = a[k+1] ;

}

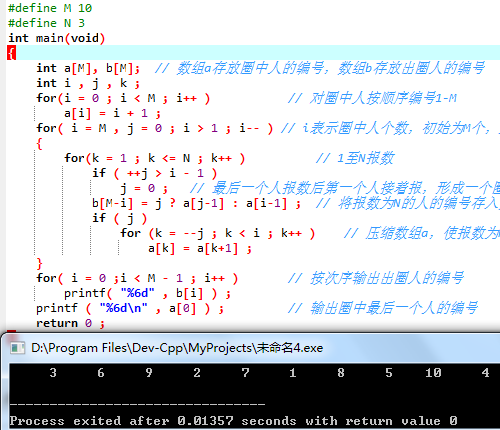
for( i = 0 ;i < M - 1 ; i++ ) // output the numbers of the circle members in order

printf( "%6d" , b[i] ) ;

printf ( "%6d\n" , a[0] ) ; // output the number of the last person in the circle

return 0 ;

}



2. In the above program, the value of the array element is used to indicate the number of the person in the circle, so every time someone leaves the circle, the array must be compressed. This algorithm is not refined enough. If the method of marking is used, that is, the corresponding array elements are marked every time someone goes out of the circle, so that the time for compressing the array can be saved, and the processing efficiency will be higher. Therefore, please use the method of marking to modify the program in ( 1 ), and make the modified program have the same function as the program in ( 1 ).

#include <stdio.h>

#define M 10

#define N 3

int main()

{

int a[M] = {0} , b[M] ;

int i , j , k ;

for ( i = M , j = 0 ; i > 0 ; i-- )

{

for ( k = 1 ; k <= N ; k++ )

{

if ( ++ j > M - 1 )

j = 0;

if ( ( a[9] == 1 && j == 0 ) || ( a [j-1] == 1 && j != 0 ) )

k -= 1 ;

}

b[Mi] = j ? j : M ;

a[j-1] = 1;

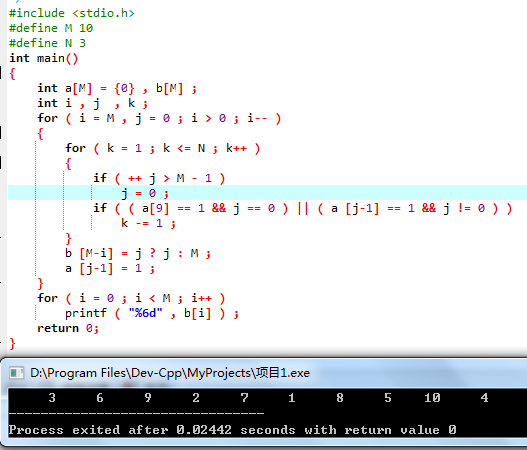
}

for ( i = 0 ; i < M ; i++ )

printf ( "%6d" , b[i] ) ;

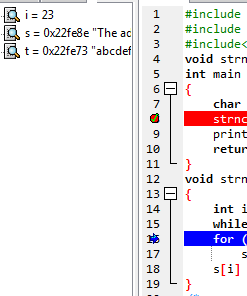
return 0;

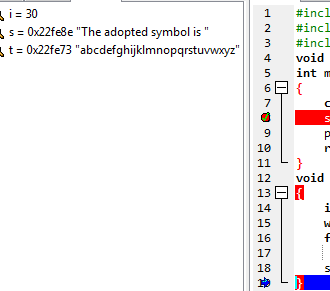
}



(3) Tracking and debugging: 2 questions

Step through the source program. After entering the function strncat, observe the expressions s, t and i. When the light bar falls on the line where the for statement is located, what is the value of i? When the light bar falls on the line where the strncat function block end mark (right curly brace }) is located, what are the values of s and t respectively?

when entering

When it ends

(2) Analyze the cause of the error of the function, eliminate the error, make the function realize the function correctly, and finally write the output result of the program.

Source program:

#include <stdio.h>

void strncat(char[],char[],int);

int main(void)

{

char a[50]="The adopted symbol is ",b[27]="abcdefghijklmnopqrstuvwxyz";

strncat(a, b, 4);

printf("%s\n",a);

return 0;

}

void strncat(char s[], char t[], int n)

{

int i = 0, j;

while(s[i++]) ;

for(j = 0; j < n && t[j];)

s[i++] = t[j++];

s[i] = '\0';

}

After modification :

#include <stdio.h>

#include <stdlib.h>

void mystrncat ( char[] , char[] , int ) ;

int main ( void )

{

char a[50] = "The adopted symbol is " , b[27] = "abcdefghijklmnopqrstuvwxyz" ;

mystrncat( a , b , 4 ) ;

printf ( "%s\n" , a ) ;

return 0 ;

}

void mystrncat ( char s[] , char t[] , int n )

{

int i = 0 , j ;

while ( s[i++] ) ;

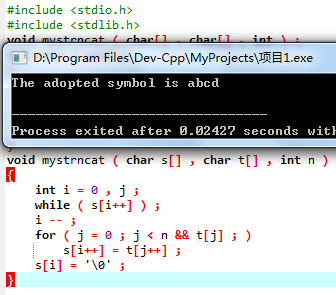
i -- ;

for ( j = 0 ; j < n && t[j] ; )

s[i++] = t[j++];

s[i] = '\0' ;

}



⑷Programming : 3 questions

1. Write a program to read data from the keyboard, ×assign values to a 3 4 matrix, find its transpose matrix, and then output the original matrix and transpose matrix.

**answer:**

1) Problem-solving ideas:

1. define matrix a[ 3][4];

2. Cyclic entry of each element of the matrix;

3. Output the original matrix, the outer loop operates on rows , and the inner loop operates on columns ;

4. Output the transposed matrix, the outer loop operates on columns , and the inner loop operates on rows ;

5. end

2 ) Source program list

int main()

{

int a[3][4] ;

char i , j ;

for ( i = 0 ; i < 3 ; i ++ ) // input matrix

for ( j = 0 ; j < 4 ; j ++ )

scanf ( "%d" , &a[i][j] ) ;

for ( i = 0 ; i < 3 ; i ++ ) // output line

{

for ( j = 0 ; j < 4 ; j ++ ) // output column

printf ( "%5d" , a[i][j] ) ;

putchar ( '\n' ) ;

}

putchar ( '\n' ) ;

for ( j = 0 ; j < 4 ; j ++ ) // output rows (columns of the original matrix)

{

for ( i = 0 ; i < 3 ; i ++ ) // output columns (rows of the original matrix)

printf ( "%5d" , a[i][j] ) ;

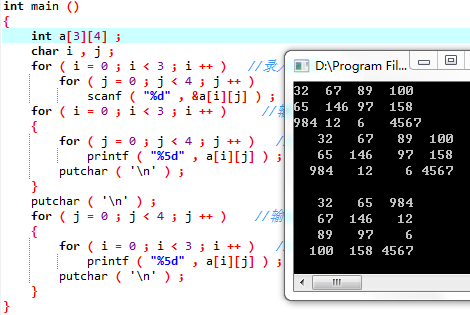
putchar ( '\n' ) ;

}

}

3 ) Test

Screenshot of the running results corresponding to the test data



2. Write a program whose functional requirements are: input an integer, convert each bit of its binary representation in memory into a corresponding digital character, store it in a character array, and then output the binary representation of the integer.

**answer:**

1) Problem-solving ideas:

1. Define INT as the int length of the current machine , and define the logical ruler mask=1 (0x01) ;

2. Input the integer number n ;

3. Loop input each integer ori[0]~ori[n-1] ;

4. Number to character array;

4.1 \_ \_ Idea: The binary representation of each data to be processed ori[i] is stored in the same order in the corresponding character array b[i] , that is, for each b[i], it is stored from the end of the string;

4.2 \_ \_ Use the logical ruler 0x01 to screen out the lowest digit of the current number and store it in b[i][INT-1] ;

4.3 The logic scale 0x01 does not move, and the a[i] type is shifted left by 1 bit ;

4.4 Enter the next cycle , and store the posted data in the lower one ;

4.5 Output the low and high obtained by the current byte

4.6 format control

5. Binary character array output, using a counter to control every eight spaces, and the last time there is no space;

6. Repeat steps 4 and 5 until all data processing is completed;

7. End \_

2 ) Program List

#include <stdio.h>

# include <limits.h>

//#define DEBUG

#define INT ( CHAR\_BIT \* sizeof ( int ) )

//The integer binary length is determined by the machine word length and the definition of int

int main()

{

int ori[100] , mask = 1 ; //mask is used to extract each binary bit of the original integer

char bi[100][INT] , i , count , j ;

unsigned short n ;

# ifdef DEBUG // check machine word length

printf ( "%d\n" , INT ) ;

#endif

scanf ( "%hu" , &n ) ;

for ( i = 0 ; i < n ; i ++ )

scanf ( "%d" , &ori[i] ) ;

# ifdef DEBUG // Check if the data input is successful

for ( i = 0 ; i < n ; i ++ )

printf ( "%d ", ori[i] ) ;

#endif

for ( i = 0 ; i < n ; i ++ )

{

count = 0;

for ( j = INT - 1 ; j >= 0 ; j -- )

{

bi[i][j] = ori[i] & mask ; // logic ruler filters out the lowest bit of binary

ori[i] >>= 1 ; // The original integer is shifted one bit to the left to prepare for the next round of screening

} // The character array stores the binary representation of the original integer forward

for ( j = 0 ; j < INT ; j ++ ) // output character array in sequence

{

printf ( "%d" , bi[i][j] ) ;

count ++ ;

if ( ! ( count % 8 ) ) // Adjust the format to facilitate checking the output

putchar ( ' ' ) ;

}

putchar ( '\n' ) ;

}

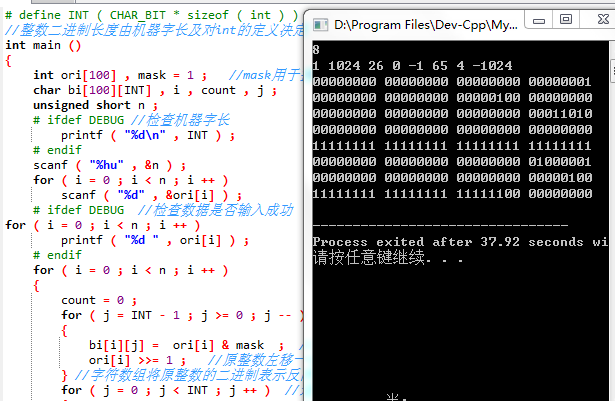
}

3 ) Test

( a ) Test Data:

n=8 ; ori[100] ={1,1024,26,0,-1,65,4,-1024}

( b ) Screenshot of the running results corresponding to the test data



3. Write a program whose functional requirements are: input the names of n students and the grades of the C language course, sort the grades from high to low, adjust the names accordingly, and output the students' names and C language grades after sorting. course grades. Then, enter a C language course grade value, and use binary search to search. If the grade is found, output the name of the classmate with the grade and the grade of the C language course; otherwise, output the prompt "not found!".

**answer:**

1) Problem-solving ideas:

1. Enter the number of grade groups n ;

2. Circularly input the student's name and corresponding grades. At this time, pay attention to the conversion between character input and number input ;

3. Pass the two arrays of name and grade into the function , and use bubble sort in the function to sort the students' grades and corresponding names in descending order ;

4. Circularly output the students' names and grades that have been sorted out ;

5. Enter the number m of grade groups to be checked;

6. Cyclic input of pending results;

7. In the loop, pass each grade to be checked and the grade array to the second function ;

8. In the function, search for the corresponding grade according to the dichotomy method, if found, return its subscript , if not found, return -1 (or any value outside the subscript value range ) ;

9. If the results to be checked are found, the corresponding student's name and grades will be output, if not found, the output will be "Not Found! ";

10. end

2 ) Source program list

void line ( char name[][20] , int score[] , int n )

{

char cpy[20];

int t , i , j ;

for ( i = 0 ; i < n ; i ++ ) // bubble sort, sort in descending order

{

\*cpy = '\0' ; // array initialization

for ( j = n - 1 ; j > i ; j -- )

{

if ( score[j] > score[j-1] )

{

t = score[j] ; // exchange scores

score[j] = score[j-1];

score[j-1] = t;

strcpy ( cpy , name[j] ) ; // exchange name

strcpy ( name[j] , name[j-1] ) ;

strcpy ( name[j-1] , cpy ) ;

}

}

}

}

int bin ( int score[] , int fin , int n ) //binary search

{

int down = 0 , up = n - 1 , mid ;

while ( down <= up )

{

mid = ( up + down ) / 2 ;

if ( fin < score[mid] ) // search in the second half

down = mid + 1;

else if ( fin > score[mid] ) // find in the first half

up = mid - 1;

else // return the target value

return mid;

}

return -1 ; // The return value cannot be the subscript of the find array

}

int main()

{

char name[100][20] , mid ;

int score[100] , find[100] , i , j , m , n ;

scanf ( "%d " , &n ) ; // Enter the number of grades and name groups

for ( i = 0 ; i < n ; i ++ )

{

j = 0;

while ( ( name[i][j] = getchar() ) != ' ' ) // Enter the name

j++;

name[i][j] = '\0' ; // convert the character array into a string for overall operation

scanf ( "%d" , &score[i] ) ; // Enter the corresponding score

getchar();

}

line ( name , score , n ) ; //score and name sorting

for ( i = 0 ; i < n ; i ++ )

{

printf ( "%-20s %d\n" , name[i] , score[i] ) ;

}

putchar ( '\n' ) ;

scanf ( "%d" , &m ) ; // Enter the number of results to be checked

for ( i = 0 ; i < m ; i ++ )

{

scanf ( "%d" , &find[i] ) ; // Enter the results to be checked

}

for( i = 0 ; i < m ; i ++ )

{

mid = bin ( score , find[i] , n ) ; //score search

if ( mid != -1 )

printf( "%-20s %d\n" , name[mid] , score [mid] ) ; //output the information corresponding to the target score

else

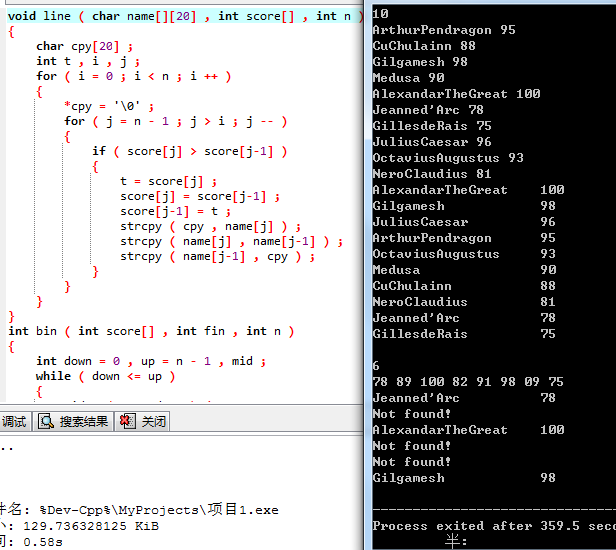
printf( "Not found!\n" ) ; // output error

}

}

3 ) Test

Screenshot of the running results corresponding to the test data



## 5.2 Optional questions

Programming: 2 questions

1. Write the function strnins(s, t, n), whose function is to insert the string t behind the nth character of the string s .

**answer:**

1) Problem-solving ideas:

1. Input the number of array groups time;

2. Read in two character strings and the insertion position ;

3. Obtain the lengths of the two strings ls and lt ;

4. The string s starts from the nth element and shifts lt backwards;

5. Insert the character string t into the space in the character string s;

6. Output the length of the new string;

7. Enter the next cycle;

8. end

2 ) Program List

#include <stdio.h>

# include <string.h>

char i ;

void strnins ( char s[] , char t[] , unsigned short n )

{

unsigned ls , lt ;

char i , j ;

ls = strlen( s ) ;

lt = strlen ( t ) ; // get the length of the string s, t

s[ls-1] = t[lt-1]= '\0' ; //s, the last element of t is replaced by 0

for ( i = ls - 2 ; i >= n ; i -- ) // The string s is staggered backwards from the nth element

s[i+lt-1] = s[i];

for ( i = n , j = 0 ; t[j] != '\0' ; i ++ , j ++ ) // insert string t into the space in string s

s[i] = t[j];

s[ls+lt-2]= '\0' ; // The new string ends with '\0'

}

int main()

{

unsigned short n , time ;

char s[500] , t[500] , blank[500] = "\0" ;

scanf ( "%hu" , &time ) ; // Number of test data sets

getchar();

for ( i = 0 ; i < time ; i ++ )

{

fgets( s , 500 , stdin ) ;

fgets( t , 500 , stdin ) ;

scanf ( "%hu" , &n ) ; // String t insertion position

getchar();

strnins ( s , t , n ) ;

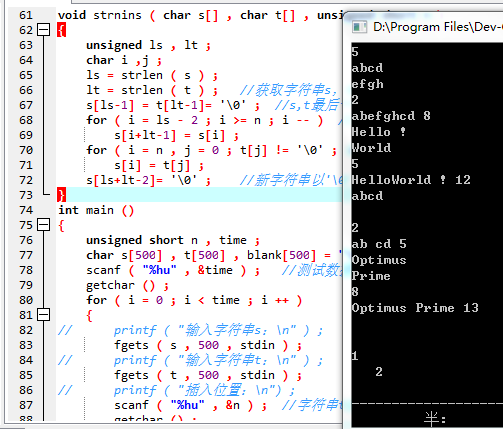
printf ( "%s %u\n" , s , strlen(s) ) ;

}

}

3 ) Test

Screenshot of the running results corresponding to the test data



2. Write a program to realize the eight queens problem , that is, put 8 queens on the 8 \*8 square chessboard , any two queens cannot be located on the same row /column/slash ( positive or negative ) , and output all possible method.

**answer:**

1) Problem-solving ideas:

1. The initial matrix is all set to 0;

2. Start to move from the first line, and use it as the basis of recursion ;

3. Search for a suitable drop point according to the rules in the second row, and if found, change the corresponding matrix element to 1 ;

4. Recurse downwards according to this idea , if there is no suitable drop point in the middle, reset the element in the previous line to 0;

5. Output the element whose value is 1 in the output row matrix after the eighth row is successfully placed ;

6. The first row moves backward one grid , recursively in a similar way ;

7. end

2 ) Program List

#include <stdio.h>

# include <limits.h>

#define N 8

\_Bool matrix[N+1][N+1] = {0};

\_Bool IsLegal ( \_Bool matrix[N + 1][N + 1] , const int i , const int j )

{

int m , n ;

// Judge whether the previous i-1 chess pieces conflict with matrix[i][j], and it is legal when i is 1

for ( m = 1 ; m <= i - 1 ; ++m )

for ( n = 1 ; n <= N ; ++n ) // Check the moves of the previous n rows, and each row actually has only one chess piece

if ( matrix[m][n] == 1 )

if ( n == j || abs ( i - m ) == abs ( j - n ) ) // on the same column/slash

return false;

return true;

}

void Print ( \_Bool matrix[N + 1][N + 1] )

{

static int count = 1 ;

int i , j ;

printf( "Case %d:\n" , count++ ) ;

for ( i = 1 ; i <= N ; i++ )

for ( j = 1 ; j <= N ; j++ )

if ( matrix[i][j] == 1 )

printf( "%d ", j ) ; // The line number is given by the output order, so only the column number is needed

putchar('\n') ;

}

void Trial ( const int i )

{

// When entering this function, i-1 chess pieces that do not attack each other have been placed in the front i-1 row of the N\*N chessboard

// Now continue to select the appropriate position for the subsequent chess pieces from the i-th row

int j;

if ( i > N ) // output a set of feasible placements

Print ( matrix ) ;

else

for ( j = 1 ; j <= N ; ++j )

{

matrix[i][j] = 1 ; //Start looking for drop points from the first column of the current row

if ( IsLegal ( matrix , i , j ) )

Trial ( i + 1 ) ; // Enter the next line after the current line succeeds

matrix[i][j] = 0;

}

}

int main ( void )

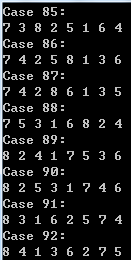
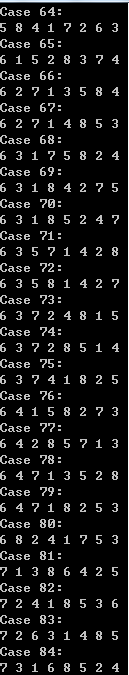
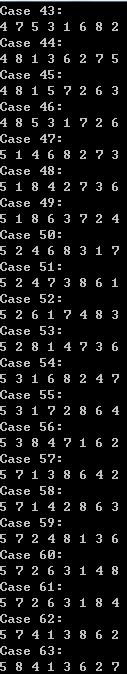
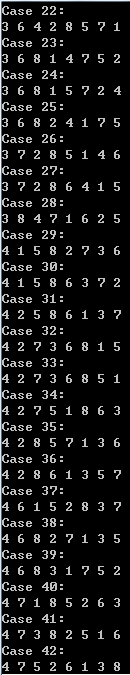
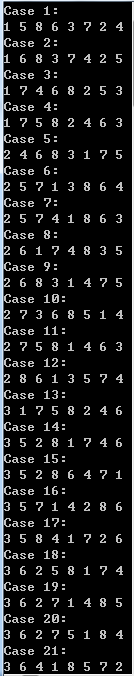
{

Trial ( 1 ) ;

return 0 ;

} 3 ) Test

Screenshot of the running results corresponding to the test data



## 5.3 Self-set questions

**(1)** Self-set experiment topic: Calculate the factorial of n .

**(2)** Purpose of the experiment: By designing the experimental program, familiarize yourself with the thinking method of processing large integers with characters.

**(3)** Algorithm steps:

1. Define a character array array with a length of 100 to store the factorial result ;

2. Loop processing input data n ;

3. Initialize the array array to 0 ;

4. Set the array[0] at the head of the string to '1' , which is the basis of factorial ;

5. Factorial ;

5. 1 sc is responsible for storing the carry value ;

5.2 Starting from i = 2, the lowest bit of the array is multiplied by i plus the last carry ( the first time is 0 ) ;

5.3 The carry is changed to sum / 10;

5.4 Sum takes the remainder of 10 , and the new result is stored in array[j];

5.5 Repeat 5.2 ~5.4 until the operation of the end of the string is completed;

5.6 Repeat 5.2 ~5.5 until i reaches the set n value ;

6. Long integers are stored in reverse order in the array , so each element of the array array is output from high to low ;

7. Repeat 3 ~ 6 until n reads into the end of the file ;

8. End ; \_

**( 4 )** Experimental procedure:

#include "stdio.h"

int main()

{

const unsigned short MAX=100 ;

char i , j , array[MAX];

unsigned short sum , sc , n ;

printf ( "n=" ) ;

while ( scanf ( "%hu" , &n ) != EOF ) // end-of-file test output

{

for ( i = 0 ; i < MAX ; i ++ )

\* ( array + i ) = 0 ; // Initialize the array each time

\* array = 1; // set factorial base

for ( i = 2 ; i <= n ; i ++ )

{

sc = 0; // responsible for carry

for ( j = 0 ; j < MAX ; j ++ )

{

sum = \* ( array + j ) \* i + sc ;

// The upper value of each bit is multiplied by the current multiplied number, and the last carry is added at the same time

sc = sum / 10 ; // store the carry value

\* ( array + j ) = sum % 10 ;

// store the remainder in the corresponding array element

}

}

printf ( "%hu!=" , n ) ;

for ( i = MAX - 1 ; i >= 0 ; i -- ) // Save the position where the output result starts

if ( \* ( array + i ) != 0 )

break;

for ( j = i ; j >= 0 ; j -- ) // output the result

printf ( "%u" , \* ( array + j ) ) ;

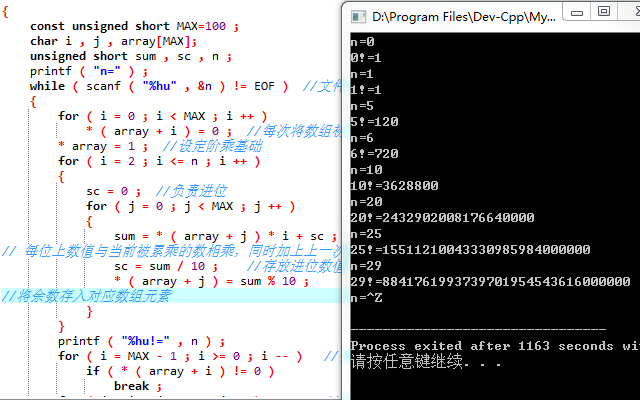
putchar ( '\n' ) ;

printf ( "n=" ) ;

}

}

**(5)** Experimental use cases: 0 1 5 6 10 25 29



**(6)** Experimental conclusion: the processing of very large integers is not suitable for reusing numeric variables, but should be processed with the help of character arrays

## 5.4 Summary

It can be found that from this time on, the program body has gradually become larger and its functions have gradually become more complex. This will be an irreversible trend in both projects and work in the future . This requires us to choose a programming thinking that suits us . Whether it is a sequential architecture that is implemented immediately after each function is written, or a layer-by-layer architecture that first writes the program body and focuses on the implementation of specific functions, we should clarify our thinking in advance. When programming, the hands should be fast and the heart should be careful, saving construction time and inspection time, so as to gradually improve their work efficiency .

It is very easy to switch between ascending order and descending order when sorting an array . You only need to think clearly about whether the subsequent item is larger or smaller than the previous item . Therefore, although there is a possibility of confusion , you only need to modify the control conditions to change it.

Although the Joseph problem was touched in the last semester , it was actually abstract, and it took a lot of work to solve it this time . The loop body in the Joseph problem is relatively abstract and the priority of the increment and decrement operators has been investigated many times . When facing such problems, you should first be calm and calm , and don’t be intimidated by unfamiliar problems. Secondly, you can patiently analyze the semantics of the program, and you can also implement it in your own pen, and get a more intuitive understanding of the code by writing and drawing step by step . Feeling , in the long run, you can enhance your ability to comprehend and visualize code when encountering unfamiliar problems .

Cooperating with auto-increment and auto-decrement embedded in the subscript can effectively reduce the amount of code , but at the same time, it is also necessary to have a deep grasp of the priority and operation sequence of the prefix and suffix auto-increment and auto-decrement . In particular, pay attention to analyze whether the subscript is out of bounds when the auto-increment operation is performed at the end of the array . If it is out of bounds , don’t forget to make up the self-decrement once, otherwise there will be a logic error, and because the error itself is hidden in the logic , it is more hidden in the back to debug Very inconvenient to find .

The transposition of the matrix is very simple to implement , but it provides a way to operate the two-dimensional/high-dimensional array non-rowwise , but you should pay attention to the writing order of the nested loop structure and the order of the subscripts , and don't make yourself around go in. From this time on , I consciously used the conditional compilation I just learned when debugging , and controlled by #ifdef DEBUG comment or not, the operation is very simple.

Finding results seems to be a huge and complicated program, but in fact, the two functions are a review of the algorithms that have been taught . Just be careful to understand the meaning of each step of the algorithm in detail and be careful not to make " penal errors " when programming . The larger the program , the harder it is to find hidden typos. Finally , you can't just be satisfied with using the algorithm in the book to solve the problem, you should try to replace the algorithm when you have time .

# 6 pointer experiment

## 6.1 Required questions

⑴ Program error correction: 1 question

Are there any errors in the following program? If so, what is the reason? If there is an error, it is required to debug and modify this sample program on the computer so that it can be executed correctly.

1 #include "stdio.h"

2 void main(void)

3 {

4 float \*p;

5 scanf("%f",p);

6 printf("%f\n",\*p);

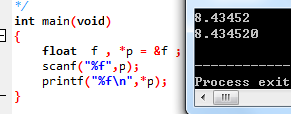
7 }

**answer:**

( 1 ) Error modification:

1) The pointer on line 4 is floating , and the correct form is:

float f , \*p = &f ;

( 2 ) Running result after error modification:

(2) Program modification and replacement: 2 questions

1. The following program calls the string copy function or string connection function through function pointers and menu selections. Please fill in appropriate expressions, statements, or code fragments in the underlined areas to complete the program.

#include "stdio.h"

#include "string.h"

void main(void)

{

char a[80],b[80],c[160],\*result=c;

int choice, i;

do {

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 exit.\n");

printf("\t\type a number (1-3) please!\n");

scanf("%d",&choice);

}while(choice<1 || choice>5);

switch(choice){

case 1:

p=strcpy;

break;

case 2:

p=strcat;

break;

case 3:

goto down;

}

getchar();

printf("input the first string please!\n");

i=0;

printf("input the second string please!\n");

i=0;

result= (a,b);

printf("the result is %s\n",result);

down:

;

}

2. In order to prevent the program from being affected by the carriage return after the input of functions such as scanf, getchar, gets, etc., please modify the program in question (1), and output the following results as required: ((input) indicates that the data is keyboard input data)

1 copy string.

2 connect string.

3 exit.

input a number (1-3) please!

2 (input)

input the first string please!

the more you learn, (enter)

input the second string please!

the more you get. (enter)

The result is the more you learn, the more you get.

**answer :**

completed program is as follows :

#include <stdio.h>

#include <string.h>

void main(void)

{

char \*(\*p) (char\*a, char\*b) ;

char a[80],b[80],c[160],\*result=c;

int choice, i;

do {

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 exit.\n");

printf("\t\type a number (1-3) please!\n");

scanf("%d",&choice);

}while(choice<1 || choice>5);

switch(choice){

case 1:

p=strcpy;

break;

case 2:

p=strcat;

break;

case 3:

goto down;

}

getchar();

printf("input the first string please!\n");

i=0;

fgets(a,80,stdin);

a[strlen(a)-1] = '\0' ;

printf("input the second string please!\n");

i=0;

fgets(b,80,stdin);

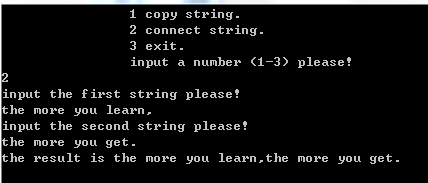
result = p(a,b);

printf("the result is %s\n",result);

down:

;

}



(3) Tracking and debugging: 2 questions

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[20],b[60]="there is a boat on the lake.";

printf("%s\n", strcpy(a,b));

}

char \*strcpy(char \*s, char \*t)

{

while(\*s++=\*t++)

;

return (s);

}

(1) Single-step execution. What is the value of s in the watch window when entering strcpy ? When returning to main, what is the value of s in the watch window?

when entering the functionC:\Users\q\AppData\Roaming\Tencent\Users\479101295\QQ\WinTemp\RichOle\S4TH]H1`AW~}%[`SI6~JD]C.png

When returning to main, there is no value of s.

(2) Eliminate errors so that the output of the program is:

there is a boat on the lake.

**answer :**

#include <stdio.h>

#include <stdlib.h>

char \*strcpy(char \*,char \*);

int main()

{

char a[60],b[60]="there is a boat on the lake.";

printf("%s\n", strcpy(a,b));

return 0;

}

char \*strcpy(char \*s, char \*t)

{

char \*cp = s;

while(\*s++=\*t++) ;

\*s='\0';

return(cp);

}

(3) Optional: Since only the value of the string pointed to by s is displayed in the watch window, and the address value stored in s is not displayed, how can we observe the change of the value of s?

Printf the address of s in the loop of the strcpy function , add a breakpoint in the while statement line or use Dev - C ++ instead

⑷Programming : 4 questions

1. A long integer variable occupies 4 bytes, and each byte is divided into high 4 bits and low 4 bits. Try to start from the high byte of the long integer variable, take out the high 4 bits and low 4 bits of each byte in turn and display them in the form of numeric characters.

**answer:**

1) Problem-solving ideas:

1. Enter the number n of data to be processed;

2. Cyclic input of data to be processed a[0]~a[n-1];

3. The pointer points to the tail ( highest bit ) of the character array with a length of 4 ;

4. Construct the logical scale 0xff , shift the long-shaped data to the left by 0, 8, 16, and 24 bits respectively and use the logical scale to filter out to obtain a char array with a length of 4;

5. Start processing binary data from a[3] (the highest bit of the original data).

4.1 \_ \_ Use the logic ruler 0x0f to screen out the lower four bits of the char type and store them in the variable low

4.2 The logic scale 0x0f does not move, and the type a[i] shifts four bits to the left ;

4.3 Screen out the upper four digits of the char type with a logic ruler , and store them in the variable high

4.4 Output the low and high obtained by the current byte

4.5 Format Control

6. Move the pointer to the left to start the next cycle;

7. End \_

2 ) Source program list

#include <stdio.h>

void litoc(long a)

{

char ch[4],\*p,low,high,i;

p=&ch[3]; // p points to the highest character array

for(i=0;i<4;i++) {

ch[i]=(a>>(CHAR\_BIT\*i))&0xff; // Scatter the binary of long type a into a character array of size 4, and store it sequentially from the low bit to the high bit of long

}

for (i=0;i<4;i++)

{

low=(\*p)&0x0f; // take the lower four bits of each byte

if(low<10)

low=low+'0'; //0~9 processing

else

low=low+'A'-10; //A~F processing

high=(\*p>>4)&0x0f; // Get the high four bits of each byte

if (high<10)

high=high+'0'; //0~9 processing

else

high=high+'A'-10; //A~F processing

printf("%c %c",high,low);

if(i!=3) {

putchar(' ') ; // format control, output without spaces after the last character

}

p--; // move the pointer back

}

putchar('\n');

}

int main()

{

long a[50];

unsigned short n, i;

scanf("%hu", &n) ; // Number of data to be processed

for(i=0;i<n;i++) {

scanf("%ld",&a[i]);

}

for(i=0;i<n;i++) {

litoc(a[i]);

}

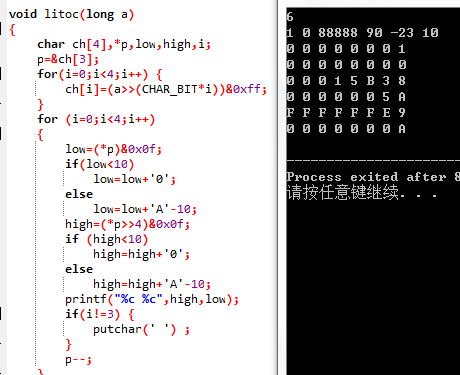
}

3 ) Test

( a ) Test Data:

1 0 88888 90 -23 10

( b ) Screenshot of the running results corresponding to the test data



2. Use the pointer array of size n to point to n lines input by the gets function, and each line does not exceed 80 characters. Write a function that compresses consecutive multiple space characters into a single space character on each line. In the calling function, output each line after compressing spaces, and blank lines will not be output.

**answer:**

1) Problem-solving ideas:

1 Input string number n, exit when n is 0;

2 define two character array a[n][N] ;

3. The fgets function loops through each character string, and correspondingly the character pointer points to the current character string ;

4. The fgets function loops through each character string;

5. The last element of the string is replaced by ' \ n' to '\0'

6. String space processing

6.1 Firstly, for each character string, screen the consecutive spaces and record the subscripts k and t of the second space (if it exists ) ;

6.2 Roll after the subscript , find the first non-empty character after it and record the subscript k; note that since the self-increment is written in the subscript, an extra step k --;

6.3 Overwrite from the second space forward with the part of the string after the non-null character .

6.4 Traverse the string until the end of the string .

6.5 Go to the next string and process it according to the above steps .

7. Circularly check the length of each string, if it is a non-empty line, it will be output normally , otherwise it will be masked;

8. End \_

2 ) Program List

#define N 80

void change (char \*p[],int n);

int main(void)

{

int i,j,n;

while(scanf("%d",&n)) { // Enter the number of string lines

if (n == 0) // termination condition

break;

n=n+1;

char a[n][N],\*p[n];

for(i=0;i<n;i++){ // Enter each string

fgets(a[i],N,stdin);

p[i]=a[i];

}

for(i=0;i<n;i++)

{

j=strlen(a[i]) ;// Set the end of the string to '\0'

a[i][j-1]='\0';

}

change(p,n);

}

}

void change(char \*p[],int n)

{

int i,j,k,t,l;

for(i=0;i<n;i++){

for(j=0;p[i][j]!='\0';j++){

if(p[i][j]==' '&&p[i][j+1]==' '){ // swallow spaces

k=j+1; // Only one space is reserved, so record the position of the second space

while(p[i][k++]==' ') ; // Record the end position of the space

k--;

t=j+1;

while(p[i][t++]=p[i][k++]); // Non-space characters move forward

}

}

if(l=strlen(p[i])>0) // swallow blank lines

printf("%s\n",p[i]);

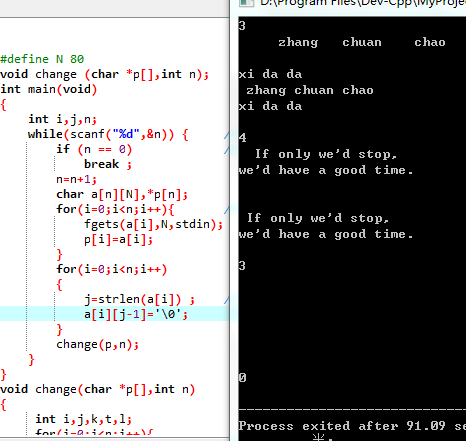
}

putchar('\n'); // format control

}

3 ) Test

Screenshot of the running results corresponding to the test data



3. Let there be N students in a certain class, and each student has taken M courses (use #define to define N and M). Enter the names of M courses, and then input the grades of the M courses taken by each of the N students in turn and store them in the corresponding array. Write the following functions:

a. Calculate the average grade of each course for each student;

b. Calculate the average grade of each course in the class;

c. Separately count the number of people whose grades are lower than the average grade of each course in the class;

d. Count the number of people who failed each course in the class and the number of people with 90 points or more (including 90 points).

Output the calculation results of the above functions in the calling function. (All pointer operations are required, and subscript operations are not allowed.)

**answer:**

1) Problem-solving ideas:

1. Input the name of each course in the same line by reading characters in a circular manner , pay attention to swallowing the separator and appending '\0' at the end of the string ;

2. Input the name of each course in the way of cyclically reading characters , and terminate with ' ' or '\ n ', pay attention to append '\0' at the end of the string;

3. Scanf loops the grades of each subject corresponding to the current student;

4. Calculate the average grade of each student in the std\_ave function, and output the calculation result at the same time ;

5. Find the average grade of each course, pay attention to the operation of the outer loop by column ( subject ) , and the operation of the inner loop by row ( student ) ;

6. In the below\_ave function, the cycle counts the number of people who are lower than the average score of each course in the class;

6.1 Pass parameters score , course, subave ; .

6.2 Define the count array count[5] to save the number of people who are lower than the average grade of the corresponding subject ; .

6.3 Operate by column (subject), compare the corresponding subject scores of each student with the average score, and count once if the student's score is low;

6.4 Output the current subject counting result num[i]; .

6.5 Enter the next cycle; .

7. Count the number of people who failed each course in the whole class in the fail function ;

7.1 Pass parameters score , course ; .

7.2 The remaining operations are the same as 6.2~6.5, but the judgment condition is changed to whether the student's score is lower than 60;

8. Count the number of people with scores above 90 (including 90) in the perfect function, the operation is the same as 7.1 ~ 7.2 , but the judgment condition is changed to whether the student's score is higher than 90;

9. End \_

2 ) Program List

#include <stdio.h>

#define N 5

#define M 5

void std\_ave(int \*s, char \*name, double stuave)

{

int sum = 0;

int i;

for (i = 0; i < N; i++)

sum += \*(s+i);

stuave = sum / 5.0;

printf("Average score of %s is %.2lf\n", name, stuave);

}

void below\_ave(int s[][M], char sub[][20], double \*subave)

{

int i, j;

int num[5] = { 0 };

for (j = 0; j < N; j++){

for (i = 0; i < M; i++){

if ((\*(\*(s + i) + j)) < \*(subave + j))

(\*(num + j)) ++;

}

}

for (i = 0; i<N; i++)

printf("Number of students lower than avg of %s is %d\n", \*(sub + i), \*(num + i));

}

void fail(int s[][M], char sub[][20])

{

int count = 0;

int i, j;

for (j = 0; j < M; j++) {

count = 0;

for (i = 0; i < N; i++){

if (\*(\*(s + i) + j) < 60) // count if the score is lower than 60

count++;

}

printf("Number of students %s fail is %d\n", \*(sub + j), count);

}

}

void perfect(int s[][M], char sub[][20])

{

int count = 0;

int i, j;

for (j = 0; j < M; j++){

count = 0;

for (i = 0; i < N; i++){

if (\*(\*(s + i) + j) >= 90)// Score is higher than 90, then count

count++;

}

printf("Number of students %s perfect is %d\n", \*(sub + j), count);

}

}

int getname(char \*s, int n)

{

char c;

int i = 0;

for (i = 0; i < n - 1 && ((c = getchar()) != EOF) && c != '\n'; i++)//If the nth character is\ni= n

\*(s + i) = c;

if (c == '\n'){

\*(s + i) = '\0';// Set '\0' at the end of the string

i++;

}

return i - 1;

}

int main()

{

char course[N][20], name[N][10];

char c;

int i, j;

int score[N][M];

double stuave[M], subave[N];

int temp = 0;

for (i = 0; i < N; i++) // course name

{

j = 0;

while ((c = getchar()) != ' '&&c != '\n'){

\*(\*(course + i) +j) = c;

j++;

}

if (c == ' ' || c == '\n'){

\*(\*(course + i) + j) = '\0';

}

}

for (i = 0; i < M; i++){

getname(\*(name + i), 10); // student name

for (j = 0; j < N; j++)

scanf("%d", &score[i][j]); // Student scores

getchar();

}

for (i = 0; i < N; i++)

std\_ave(\*(score + i), \*(name + i), \*(stuave + i)); //The average grade of each student

for (j = 0; j < N; j++){

temp = 0;

for (i = 0; i < M; i++){

temp += \*(\*(score + i) + j); // Average grade for each course

}

\*(subave + j) = temp / 5.0;

printf("Average score of %s is %.2lf\n", \*(course + j), \*(subave + j));

}

below\_ave(score, course, subave);// The number of people who are lower than the average score of each course in the class

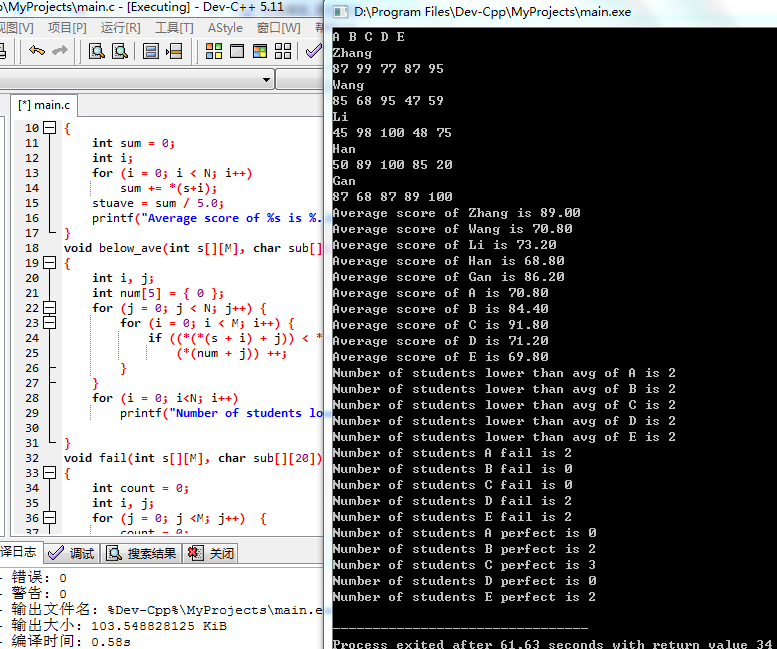
fail(score, course); // The number of people who failed each course

perfect(score, course); // The number of people with 90 points or more (including 90 points)

}

3 ) Test

Screenshot of the running results corresponding to the test data



4. Write a program that inputs n integers and outputs them after sorting. The principle of sorting is determined by the optional parameter -d on the command line. When the parameter -d is present, it is sorted in descending order, otherwise it is sorted in ascending order. It is required to define the sorting algorithm as a function, and use the pointer to the function to make the function realize ascending or descending sorting.

**answer:**

1) Problem-solving ideas:

1 Call the system environment , enter the program address ( and command line parameters ), and then judge the sorting order, see the topic for details;

2. If the number of parameters or the second parameter is not "-d" , an error message will be output and popped up;

3. Input data number n ;

4. Cyclic entry of each element a[0]~a[n-1] of the array ;

6. Pass the array, the number of elements, and the tag value into the sort function for sorting;

7. Use selection sorting to organize the array , and the sorting method is determined by whether there is a tag value determined by "-d " ;

8. Output the sorted array;

9. End \_

2 ) Program List

#include <stdio.h>

void sort(int a[], unsigned short x, char flag)

{

int temp;

char i, j;

for(i = 0; i < x; i ++)

{

for(j = 0; j < x - i - 1; j ++)

{

if(flag ? (a[j] < a[j+1] ) : (a[j] > a[j+1])) //The judgment condition is the flag value determined by the command line parameters

{

temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

}

int main(int argc,char \*argv[]) //main parameters are given by the system

{

unsigned short n ;

char j = 1, i;

if(argc > 2) // The number of parameters does not match

{

printf("Error!\n");

return -1;

}

else if(argc == 2) // parameter content does not match

{

if(\*argv[1] != '-' || \*(argv[1] + 1) != 'd')

printf("Error!\n");

return -1;

}

else j = 0;

int array[50];

scanf("%hu", &n) ; // Determine the number of data groups

for(i = 0; i < n; i ++)

scanf("%d", &array[i]); // input array

sort(array, n, j); // sort by category

for(i = 0; i < n; i ++)

{

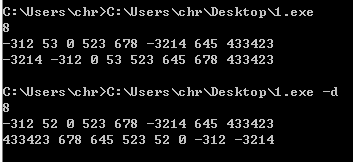
printf("%d ", array[i]);

}

}

3 ) Test

Screenshot of the running results corresponding to the test data



## 6.2 Optional questions

Programming: 2 questions

1. There are data a and b with N-digit integers and M-digit decimals (N=20, M=10). Program to calculate a+b and output the result.

Such as: 12345678912345678912.1234567891 + 98765432109876543210.0123456789

**answer:**

1) Problem-solving ideas:

1. Define the length M of the integer part and the length N of the fractional part (Note : This is contrary to the requirements of the title );

2Define the character arrays of the integer part and the decimal part of the three super long integers a, b, c , the length is M/N +2 because one bit stores '\ 0 ', and the other bit of c stores the carry, a, b's The other bit stores the suffix carriage return when reading in;

3. Enter the integer part of a ,;

3.1 The fgets function reads the integer part of a ;

3.2 Use to obtain its length lia , and replace the carriage return character at the end of the string with '\0 ' ;

3.3 Align the long integers with high bits in the array to facilitate subsequent calculations ;

3.4 Traverse the pointer to find the highest bit of the digital part ;

3.5 The part of the string that is not filled (for the integer part is the high bit) is filled with '0' to facilitate subsequent calculations;

4. Enter the decimal part of a ,;

4.1 The fgets function reads the fractional part of a ;

4.2 obtain its length l n a by using ;

4.3 The part of the string that is not filled (the low bit for the decimal part) is filled with '0' to facilitate subsequent calculations;

5. Input the positive part of b, the method is the same as 3.1-3.5 , and the length is stored in lib ;

6. Input the decimal part of b, the method is the same as 4.1-4.3 , and the length is stored in l n b ;

7. Under DEBUG , enter the super long integers a and b in a unified format for easy inspection ;

8 .a , b The decimal part is added (ad num function is realized ) ;

8.1 The larger one among lna and lnb is used as the string length of numc , which is recorded as lnc ;

8.2 The decimal part array numc string of the sum is set with '\0' at the end ;

8.3 Starting from the lowest bit given by lnc , convert the numbers at the corresponding positions of numa and numb into integers , add them to the last save (the first save is 0) , and store the result in sum;

8.4 Determine the size relationship between sum and 10. If it is less than 10, the sum will be converted into a character number and stored in the corresponding digit of numc . Otherwise, sum-10 will be performed first and then subsequent operations will be performed , and the carry will be stored in save;

8.5 The highest bit of the decimal part ends, and the remaining carry save is sent back to the main function ;

8.6 When DEBUG is defined , the decimal part of output c is easy to check ;

9. Add the integer parts of a and b and adjust the format ( implemented by ad int function );

9.1 The carry returned by the adnum function is passed to ad int together with the integer part of a and b ;

9.2 lia, the larger one in lib is the possible string length of int c (considering the carry) , recorded as l i c ;

9.3 The integer part array int c string of the sum is set with '\0' at the end ;

9.4 The integer addition method of a and b is the same as 8.3-8.4

9.5 To the end of the highest bit (lic) of the integer part;

9.6 The highest bit ( string head ) is changed to the symbolization of the last save , and it is also considered a, b There is at least one special case where the integer part occupies M bits ;

9.7 When DEBUG is defined , the integer part of output c is easy to check ;

9.8 format processing ;

9.8.1 The pointer points to the highest bit of numc ( near the string head ) ;

9.8.3 Shift the character string to the left as a whole (including '\0'), so that the integer part of c is left -aligned and stored in int c ;

9.9 Under the definition of DEBUG , output the processed integer part in c format for easy inspection ;

10. Respectively output the integer part and the decimal part of the super long integer c ;

11. End \_

2 ) Program List

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define M 20

#define N 10

//#define DEBUG

unsigned short lia, lib, lna, lnb, lic, lnc ;

void get(char \*inta, char \*intb, char \*numa, char \*numb) /\* input floating point numbers a, b and store them in a unified format\*/

{

char i, \*p ;

printf("input the integer part of a :") ;

fgets(inta, M+2, stdin) ;

lia = strlen(inta)-1 ; /\* get the length of the integer part\*/

\*(inta + M) = '\0' ; /\* end position '\0'\*/

/\* Number strings are right aligned in the array\*/

for (i = lia - 1, p = inta + M - 1; i > -1; i --, \*p --) {

\*p = \*(inta + i) ;

}

/\* fill non-numeric parts with 0 \*/

for(i = 0; i < M - lia; i ++) {

\*(inta + i) = '0';

}

printf("input the numeric part of a :") ;

fgets(numa, N+2, stdin) ;

lna = strlen(numa)-1 ; /\* get the length of the decimal part\*/

for (i = lna; i < N; i ++) { /\* non-number parts are filled with 0 \*/

\*(numa + i) = '0';

}

\*(numa + N) = '\0' ;

printf("input the integer part of b :") ;

fgets(intb, M+2, stdin) ;

lib = strlen(intb)-1;

\*(intb + M) = '\0';

for (i = lib - 1, p = intb + M - 1; i > -1; i --, \*p --) {

\*p = \*(intb +i) ;

}

for(i = 0; i < M - lib; i ++) {

intb[i] = '0';

}

printf("input the numeric part of b :") ;

fgets(numb, N+2, stdin) ;

lnb = strlen(numb)-1;

for (i = lnb; i < N; i ++) {

\*(numb + i) = '0';

}

\*(numb + N) = '\0' ;

#ifdef DEBUG

printf("%s.%s\n%s.%s\n", inta, numa, intb, numb) ;

#endif

}

char adnum(char \*numa, char\* numb, char \*numc)

{

char save = 0, i, sum;

lnc = lna > lnb ? lna : lnb ; /\* Get the length of the number string to be added\*/

\*(numc + lnc) = '\0' ; /\* end the string with '\0'\*/

for(i = lnc - 1; i >= 0 ; i --) {

sum = \*(numa + i) - '0' + \*(numb + i) - '0' + save ;/\* Convert the same digit of the string into a number and add it\*/

if(sum >= 10) { /\* store the unit digit of sum in the same position in sumc, and assign the carry to save\*/

\*(numc + i) = sum - 10 + '0';

save = 1;

}

else {

\*(numc + i) = sum + '0';

save = 0;

}

}

#ifdef DEBUG

printf("%s", numc);

#endif

return save ; /\* Return the carry of the highest digit of the decimal part\*/

}

void adint(char \*inta, char\* intb, char \*intc, char get)

{

char save = get, i, sum, \*p, add;

lic = lia > lib ? lia : lib ; /\* Get the length of the number string to be added\*/

\*(intc + M + 1) = '\0' ; /\* end of string (the last element) is set to '\0' \*/

for(i = M - 1; i >= 0 ; i --) { /\* The operation is the same as adding the decimal part\*/

sum = \*(inta + i) - '0' + \*(intb + i) - '0' + save;

if(sum >= 10) {

\*(intc + i + 1) = sum - 10 + '0';

save = 1;

}

else {

\*(intc + i + 1) = sum + '0';

save = 0;

}

\*intc = save + '0';

if(i == M + 1 - lic) { /\* Get the last carry of the number string\*/

add = save;

}

}

#ifdef DEBUG

printf("%s\n", intc) ;

#endif

p = &intc[M+1-(lic+add)] ; /\* Format processing, swallow the high-order 0 of the number string\*/

for (i = 0; p < intc + M + 2; i ++, p ++) {

\*(intc + i) = \*p;

}

#ifdef DEBUG

printf("%s\n", intc) ;

#endif

}

int main()

{

/\*int stores the integer part, num stores the real part\*/

char inta[M+2], numa[N+2], intb[M+2], numb[N+2], intc[M+2], numc[N+2], i, j, extra ;

i = 0;

get(inta, intb, numa, numb) ; /\* Input floating point numbers a, b and store them in a unified format\*/

extra = adnum(numa, numb, numc) ;/\* Add the decimal part, return the carry\*/

adint(inta, intb, intc, extra) ;/\* Add the integer part\*/

printf("the sum of a and b is %s.%s\n", intc, numc) ;

}

3 ) Test

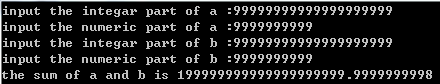
( a ) Test Data:

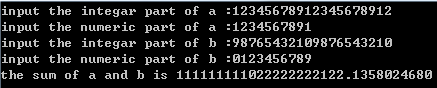
①a=99999999999999999999.9999999999; b=99999999999999999999.9999999999

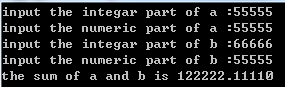
② a= 12345678912345678912.1234567891; b=98765432109876543210.0123456789

③ a=55555.55555; b=66666.55555

( b ) Screenshot of the running results corresponding to the test data







2. Write a program that uses the complex declaration char \*(\*p[2])(const char \*, const char \*);

Tip: The elements in p can be function names such as strcmp and strstr.

**answer:**

1) Problem-solving ideas:

1. Define the string comparison function tostrcmp, and the judgment substring function to strcat , both functions return the pointer to the corresponding library function , note that the return value of strcmp in the header file <string.h> is int type , so (char \*) is required do a forced conversion;

2. Declare an array of function pointers char \*(\*p[2])(const char \*, const char \*) and initialize them as tostrcmp and to strcat respectively ;

3. Use fgets to input two strings , pay attention to clear the newline character at the end of the string to prevent interference during comparison ;

4. Enter the option n to call the function;

5. Use ( char \* ) re to store the pointer value returned by the called function;

6. Switch structure output information ;

6.1 When n is 0, call tostrcmp, judge whether the two strings are the same according to the returned re value and decide to output information ;

6.2 When n is 1, call tostrstr, judge whether b is a substring of a according to the returned re value , and determine the output information ;

7. End \_

2 ) Program List

#include <stdio.h>

#include <string.h>

char \*tostrcmp(const char \*a, const char \*b) /\* call strcmp of the system\*/

{

return (char \*)strcmp(a,b) ;

}

char \*tostrstr(const char \*s, const char \*t) /\* call strstr of the system\*/

{

return strstr(s,t) ;

}

char \*(\*p[2])(const char \*, const char \*) ;

int main()

{

char a[100], b[100], \*re ;

unsigned short n ;

p[0] = tostrcmp;

p[1] = tostrstr;

fgets(a, 50, stdin) ;

fgets(b, 50, stdin) ;

a[strlen(a)-1] = b[strlen(b)-1] = 0 ; /\* Eliminate the newline at the end of the string\*/

scanf("%hu", &n) ; /\* function call option\*/

re = p[n](a, b) ;

switch(n) {

case 0:

if(!re) {

printf("a and b are the same!\n") ;

}

else {

printf("a and b are different!\n") ;

}

break;

case 1:

if(re)

printf("b is included in a\n") ;

else

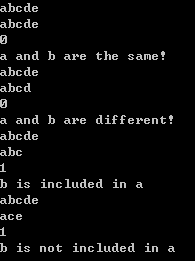
printf("b is not included in a\n") ;

}

}

3 ) Test

Screenshot of the running results corresponding to the test data



## 6.3 Self-made questions

**(1)** Self-set experiment topic: Function pointers are used as function parameters to realize four arithmetic operations

**(2)** Purpose of the experiment: To practice the use of function pointers by designing experimental programs .

**(3)** Algorithm description :

1. Design the directory function float mathfunc(float(\*p)(float, float), float para1, float para2) ; pay attention to the function declaration and variables as parameters, and the return value is a function in addition, subtraction, multiplication and division;

2. Design add (addition), min (subtraction), tim ( multiplication ) , div ( division ) four functions according to the four arithmetic operations ;

3. Define two float type variables a , b, note that none of them can be 0 because of the existence of division ;

4. Output the results of the four arithmetic operations separately , and pay attention to the format of the function writing corresponding to the directory function ;

5. end \_

**(4)** Experimental procedure:

/\* function name is passed as parameter \*/

float mathfunc(float(\*p)(float, float), float para1, float para2) {

return(\*p)(para1, para2) ; /\* return function\*/

}

float add(float f1, float f2) {

return f1 + f2 ;

}

float min(float f1, float f2) {

return f1 - f2 ;

}

float tim(float f1, float f2) {

return f1 \* f2 ;

}

float div(float f1, float f2) {

return f1 / f2 ;

}

int main () {

float a = 1.5, b = 2.5;

printf("a+b=%f\n", mathfunc(add, a, b)) ;

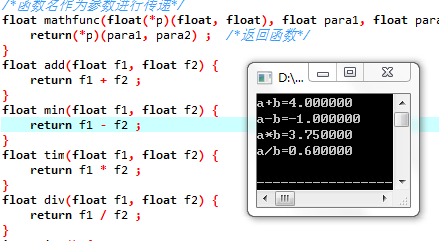
printf("ab=%f\n", mathfunc(min, a, b)) ;

printf("a\*b=%f\n", mathfunc(tim, a, b)) ;

printf("a/b=%f\n", mathfunc(div, a, b)) ;

}

**(5)** Experimental use case: a=1.5, b=2.5



**(6)** Experimental conclusion: Passing the function name as a parameter provides another idea of function overloading.

## 6.4 Summary

This time, it can be clearly felt that the task has become more difficult , and this " difficulty " is caused by two aspects . One is that this test has caught up with the three exams in the 13th and 14th weeks . The pressure of review will make people want to get out of the usual homework and prepare for the exam as soon as possible; the second is that the knowledge points themselves are more difficult . It is more difficult when you get started for the first time , and there is a greater chance of making mistakes when programming, and you will find the location and cause of the error in a short time ; People feel irritable and even form positive feedback.

The most basic pointer definition declaration is also prone to errors . It should be noted that the wording of declaring simultaneous initialization is different from that of first declaring and then initializing, and there is a difference of a pointer specifier "\*" . This time, the space processing is further improved , and the requirement of shielding blank lines is added while storing multiple lines of statements at the same time . There are two options here : one is to directly delete this blank line in the memory and let the subsequent sentence lines move forward in sequence, and the other is not to output when the current line is judged to be a blank line . Obviously , the latter is much easier to implement, and it is completely feasible when the program body basically consumes a lot of memory. You should pay attention to this method when programming in the future .

What stuck me this time was the entry of grades and the two optional questions.

Grade entry involves the mixed input of strings and numbers , which involves the understanding and distinction of character string input, scanf function input , and fgets function input . Of course, unfortunately, I wasted a lot of time in the data entry process because I didn't distinguish these input methods clearly before. Of course, after constant trial and error and comparison , this problem was finally solved. In addition, the diversity of function parameter passing values has been greatly improved this time, and there is also the centralized use of array pointer representation, and the amount of single code has reached a new level . It is also a test of skill to calmly figure out each problem .

In addition to the amount of code in the calculator itself , there is also the problem of distinguishing numbers from other characters when inputting because of the need to store numbers in strings. In addition , there are various functions such as carry, addition of characters to numbers , and alignment of digits. detail. In fact, simply completing the function does not require a lot of code, but further format requirements such as deleting the '0' in the string and the removal of the last zero after the decimal point ( there is no such function in this program ) The " culprit " of increasing the amount of code . The declaration of the function pointer is simply difficult in its correct declaration and application, and the difficulty of the program itself is not high.

In view of the fact that pointers are frequent visitors in future programming , it is necessary to master and use some knowledge of pointers , but this time the experimental questions are far from enough, and multi-base codes are still required after class.

# 7 Structure and Joint Experiment

## 7.1 Required Questions

(1) Expression evaluation and program verification: 1 question

With instructions:

char u[]=" UVWXYZ ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

}a[]={{1 1 , ˊAˊ ,u},{10 0 , ˊBˊ ,v}},\*p=a;

Please first calculate the value of the following expression yourself, and then verify it through programming calculations. (Each expression has nothing to do with each other)

|  |  |  |  |
| --- | --- | --- | --- |
| **serial number** | **expression** | **Calculated** | **verification value** |
| 1 | (++p)->x | 100 | 100 |
| 2 | p++, p->c | ( address), B | 00403010,A |
| 3 | \*p++->t , \*p->t | U, x | U, u |
| 4 | \*(++p)->t | x | x |
| 5 | \*++p->t | V | V |
| 6 | ++\*p->t | V | V |

(2) Program modification and replacement: 2 questions

Given a batch of integers, with 0 as the end mark and not as a node, build it into a first-in-first-out linked list. The pointer pointer of the first-in first-out linked list always points to the first created node (chain head), and the first node is built The point points to the post-build node, which is always the tail node.

1. What kind of errors exist in the source program (observe the execution result first)? Modify and debug the program so that it can correctly complete the specified tasks.

The source program is as follows:

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* data field\*/

struct s\_list \*next; /\* pointer field\*/

} ;

void create\_list (struct s\_list \*headp, int \*p);

void main(void)

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0}; /\* 0 is the end mark\*/

create\_list(head,s); /\* Create a new linked list\*/

p=head; /\*traversal pointer p points to the chain head\*/

while(p){

printf("%d\t",p->data); /\* Output the value of the data field\*/

p=p->next; /\*traversal pointer p points to the next node\*/

}

printf("\n");

}

void create\_list(struct s\_list \*headp, int \*p)

{

struct s\_list \* loc\_head=NULL,\*tail;

if(p[0]==0) /\* equivalent to \*p==0 \*/

;

else { /\* loc\_head points to the first dynamically allocated node \*/

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++; /\* assign value to data field\*/

tail=loc\_head; /\* tail points to the first node\*/

while(\*p){ /\* The pointer field of the node pointed to by tail points to the dynamically created node\* /

tail->next=(struct s\_list \*)malloc(sizeof(struct s\_list));

tail=tail->next; /\* tail points to the newly created node\*/

tail->data=\*p++; /\* assign value to the data field of the newly created node\*/

}

tail->next=NULL; /\* Assign NULL value to the pointer field \*/

}

headp=loc\_head; /\* Make the head pointer headp point to the newly created linked list \*/

}

**answer :**

The original program has no output , and the replaced program is as follows :

struct s\_list {

int data; /\* data field\*/

struct s\_list \*next; /\* pointer field\*/

} ;

struct s\_list \*create\_list (struct s\_list \*headp,int \*p);

void main(void)

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0}; /\* 0 is the end mark\*/

p = create\_list(head,s); /\* Create a new linked list\*/

// p=head; /\* traversal pointer p points to the chain head\*/

while(p){

printf("%d\t",p->data); /\* output the value of the data field\*/

p=p->next; /\* traversal pointer p points to the next node\*/

}

printf("\n");

}

struct s\_list \*create\_list(struct s\_list \*headp, int \*p)

{

struct s\_list \* loc\_head=NULL,\*tail;

if(p[0]==0) /\* equivalent to \*p==0 \*/

;

else { /\* loc\_head points to the first dynamically allocated node \*/

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++; /\* assign value to data field\*/

tail=loc\_head; /\* tail points to the first node\*/

while(\*p){ /\* The pointer field of the node pointed to by tail points to the dynamically created node\*/

tail->next=(struct s\_list \*)malloc(sizeof(struct s\_list));

tail=tail->next; /\* tail points to the newly created node\*/

tail->data=\*p++; /\* Assign a value to the data field of the newly created node\*/

}

tail->next=NULL; /\* Assign NULL value to the pointer field\*/

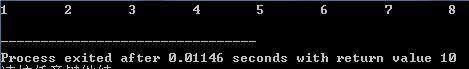
}

headp=loc\_head; /\* Make the head pointer headp point to the newly created linked list\*/

return headp;

}

The program running results are as follows, the program is correct.



(2) Modify and replace the create\_list function to build it into a last-in-first-out linked list. The head pointer of the last-in first-out linked list always points to the last created node (chain head), and the later-built node points to the first-built node, and the first-built node A point is always a tail node.

**answer :**

The replaced program looks like this :

struct s\_list{

int data;

struct s\_list \*next;

};

void creat\_list(struct s\_list \*\*headp, int \*p);

int main()

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0};

creat\_list(&head,s);

p=head;

while(p){

printf("%d\t",p->data);

p=p->next; }

printf("\n");

return 0;

}

void creat\_list(struct s\_list \*\*headp, int \*p)

{

struct s\_list\*loc\_head=NULL,\*tail;

struct s\_list\*temp;

if(p[0]==0) ;

else {

loc\_head=(struct s\_list\*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++;

tail=loc\_head;

while(\*p){

temp=(struct s\_list\*)malloc(sizeof(struct s\_list));

temp->next=loc\_head;

loc\_head=temp;

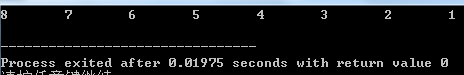
loc\_head->data = \*p++; }

tail->next=NULL; }

\*headp=loc\_head;

}

The program running results are as follows, the program is correct.



(3) Programming: 2 questions

1. Design a field structure struct bits , which declares an 8-bit unsigned byte as 8 fields from the lowest bit to the highest bit, and each field is bit0, bit1, … , bit7 in turn, and bit0 has the highest priority. Design 8 functions at the same time, the i-th function takes biti (i=0,1,2, … ,7) as a parameter, and outputs the value of biti in the function body. Store the names of 8 functions into a function pointer array p\_fun. If bit0 is 1, call the function pointed to by p\_fun[0]. If more than one bit in the struct bits is 1, the function pointed to by the corresponding element in the function pointer array p\_fun is called sequentially according to the priority from high to low. The 0th function among the 8 functions can be designed as:

void f0( struct bits b)

{

P rintf( " the function %d is called!\n " ,b);

}

**answer:**

1) Problem-solving ideas:

1. Design the field structure struct ISR\_BITS , which contains unsigned type bit0-bit7 (each occupying 1 bit ) and rsv8 (aligning the high bit of unsigned short ) and declare the variable bit;

2. Design the joint structure union ISR\_REG, which contains unsigned short type variable a and struct ISR\_BITS type variable bit;

3. Define void-type functions f0-f8 for outputting information ;

4. Define an array of function pointers with a length of 8 and assign values to f0-f8 one by one;

5. Input the number n of data to be processed ;

6. Cyclic input of each data to be processed ;

7. Process each data;

7.1 Store the array in the unsigned short type of the union structure ;

7.2 Use the struct structure to access and process the data in the union ;

7.3 Determine whether bit 0 - bit8 is 1, if it is 1, enter the corresponding output function;

8. End \_

2 ) Source program list

struct ISR\_BITS { // field structure

unsigned bit0: 1, bit1: 1, bit2: 1, bit3: 1, bit4: 1, bit5: 1, bit6: 1, bit7: 1, rsv: 8 ;

} bit;

union ISR\_REG { //Field structure read in indirectly

unsigned short a ;

struct ISR\_BITS bit;

}temp;

// output function group

void f0(struct ISR\_BITS b) {

printf("the 1 function %d is called!\n", b.bit0); }

void f1(struct ISR\_BITS b) {

printf("the 2 function %d is called!\n", b.bit1);}

void f2(struct ISR\_BITS b) {

printf("the 3 function %d is called!\n", b.bit2); }

void f3(struct ISR\_BITS b){

printf("the 4 function %d is called!\n", b.bit3);}

void f4(struct ISR\_BITS b){

printf("the 5 function %d is called!\n", b.bit4);}

void f5(struct ISR\_BITS b){

printf("the 6 function %d is called!\n", b.bit5);}

void f6(struct ISR\_BITS b){

printf("the 7 function %d is called!\n", b.bit6);}

void f7(struct ISR\_BITS b){

printf("the 8 function %d is called!\n", b.bit7);}

int main()

{

void (\*p\_fun[8])(struct ISR\_BITS b);

unsigned short n, num[50] ;

char i ;

scanf("%hu", &n) ;

for(i = 0; i < n ; i ++) {

scanf("%hu", &num[i]) ;

}

for(i = 0; i < n; i ++) {

printf("%hu:\n", num[i]) ;

temp.a = num[i];

p\_fun[0] = f0; // function pointer assignment

p\_fun[1] = f1;

p\_fun[2] = f2;

p\_fun[3] = f3;

p\_fun[4] = f4;

p\_fun[5] = f5;

p\_fun[6] = f6;

p\_fun[7] = f7;

if(temp.bit.bit0) p\_fun[0](temp.bit); // access between function pointers

if(temp.bit.bit1) p\_fun[1](temp.bit);

if(temp.bit.bit2) p\_fun[2](temp.bit);

if(temp.bit.bit3) p\_fun[3](temp.bit);

if(temp.bit.bit4) p\_fun[4](temp.bit);

if(temp.bit.bit5) p\_fun[5](temp.bit);

if(temp.bit.bit6) p\_fun[6](temp.bit);

if(temp.bit.bit7) p\_fun[7](temp.bit);

putchar('\n') ;

}

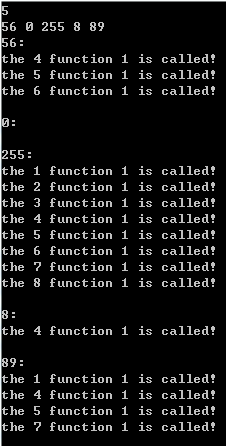
}

3 ) Test

( a ) Test Data:

56; 0; 255; 8; 89

( b ) Screenshot of the running results corresponding to the test data



2. Create a class report card with a one-way linked list, including each student's student number, name, English, advanced mathematics, general physics, and C language programming. Use functional programming to achieve the following functions:

(1) Enter the various information of each student.

(2) Output various information of each student.

(3) Modify the content of the specified data item for the specified student.

(4) Calculate the average grade of each student (reserve 2 decimal places).

(5) Output the student ID, name, total grade and average grade of the four courses of each student.

( Cooperate with the first and second optional questions )

**answer:**

1) Problem-solving ideas:

1. Define the student information structure struct student, including num [15] (student number), name[20] (name), eng ( English score ) , cal (mathematics score), phy (physics score), cpp (C language score ), sum ( total score ) , avg ( average score ) , \*next ( self-reference ) , and create corresponding traversal pointers \* head, \*tail at the same time , pointing to the head of the linked list ;

2. Define the grade modification structure struct alter, including num15 (student number), sub[8] (subject to be modified), score ( new grade ) , \*next (self-reference), and create corresponding traversal pointers \* fore,\*aft , Point to the head of the linked list;

3. Define the header structure struct head, including id [3] (ID), name [5] (name), eng [8] (E nglish ), cal [5] ( Maths ), phy [8] (P hysics ), c [2] (C), sum [4] (SUM), avg [4] (AVG) ;

4. Define the number of records n ,;

5. Enter the number of records n ;

6. Cyclic entry of each struct student record

6.1 The scanf function inputs each member of the structure, pay attention to swallow the carriage return;

6.2 Create a new linked list backwards , and the location is pointed by the self-reference pointer of the current linked list section ;

6.3 The self-reference pointer of the new linked list section is empty ;

7. Output the table header , including student number, name, and grades of each subject ;

8 Loop out each record, and at the same time traverse the pointer to the next link list , pay attention to add a blank line after exiting the loop ;

9. Enter the number of subjects to be modified m;

10. Cyclic input of each struct alter record, the method and precautions are the same as 6.1 -6.3;

11. Traversing the pointer tail, aft refers back to the corresponding linked list head;

12. Circularly compare and enter the new score record with the original record ;

12.1 Pass the new grade currently pointed to by the traversal pointer , struct student head pointer and table header pointer \*head to the comparison function amend (struct student \*idv , struct alter \*stu, struct title \*head);

12.2 compare the student number of struct alter with the student number of struct student one by one, if they match , enter 12.3, if they do not match, enter 12.5 ;

12.3 Compare the sub of struct alter with the subject names of struct title one by one, if it matches , enter 12.4, if it does not match, enter 12.5 ;

12.4 Replace the corresponding old grades of struct student with the new grades of struct alter (through the one-to-one correspondence between the member names of struct title and the member names of struct student ) ;

12.5 . After the comparison/modification is over, exit the function ;

12.6. Traverse the pointer backwards;

12.7 Repeat 12.1-12.5 until the traversal pointer is empty;

13. Same as 7;

14. Traverse the pointer tail to point back to the corresponding linked list head;

15. Statistical results ;

15.1 Tail and table header pointer h are passed to the performance statistics function stat ;

15.2 Output table header , including student number, name, total score and average score;

15.3 Calculate and output the total grade and average grade of each student in a loop , and push backward according to 6.2 and 6.3 ;

16. Output the updated records , pay attention to the attached header and format control ;

17. Same as 14;

18-0 . Sort by grade point average (swap pointer field ) ;

18-1 . Sort by average grade (exchange pointer field );

19. Same as 15.2 ;

20. Circularly output the sorted student information, including student number , name, and average grade , and push backward according to 6.2 and 6.3 ;

21. End \_

2 ) Program List

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct student { // Student number + name + scores of each subject + total score + average score

char num[15], name[20];

float eng, cal, phy, cpp, sum, avg;

struct student \*next ;

};

struct alter { // Student number + subjects to be modified + new grades

char num[15], sub[8];

float score;

struct alter \*next ;

};

struct title { // header

char id[3], name[5], eng[8], cal[5], phy[8], c[2], sum[4], avg[4] ;

}head={"ID", "Name", "English", "Math", "Physics", "C", "SUM", "AVG"}, \*h = &head ;

unsigned short n ;

//Modify grades by student number and subject

void amend(struct student \*idv, struct alter \*stu, struct title \*head)

{

while(idv) {

if(!strcmp(idv->num, stu->num)) { // Student number judgment

if(!strcmp(head->eng, stu->sub)) {// subject judgment

idv->eng = stu->score;

}

else if (!strcmp(head->cal, stu->sub)) {

idv->cal = stu->score;

}

else if (!strcmp(head->phy, stu->sub)) {

idv->phy = stu->score;

}

else if (!strcmp(head->c, stu->sub)) {

idv->cpp = stu->score;

}

}

idv = idv->next;

}

}

void stat(struct student \*stu, struct title \*head) // Statistics total score and average score

{

struct student \*save = stu;

printf("\nSumAndAvg:\n%-15s%-20s%-10s%-10s\n", head->id, head->name, head->sum, head->avg) ;

while(stu->next) {

stu->sum = stu->cal + stu->cpp + stu->eng + stu->phy ;

stu->avg = stu->sum / 4;

printf("%-15s%-20s%-10.2f%-10.2f\n", stu->num, stu->name, stu->sum, stu->avg) ;

stu = stu->next;

}

}

void sort(struct student \*stu, struct title \*head) //Sort by average grade (exchange data fields)

{

struct student \*p1 = stu, \*p2 ;

char i, j, temp[20] = "";

unsigned short len = 0;

printf("\nSort:\n%-15s%-20s%-10s\n", head->id, head->name, head->avg) ;

float t;

while(p1) {

len ++ ;

p1 = p1->next;

}

for(i = 0, p1 = stu; i < len - 1; i ++, p1=p1->next ) {

for(j = i + 1, p2 = p1->next; j < len - 1; j ++, p2 = p2->next) {

if(p1->avg > p2->avg) {

t = p1->avg ;p1->avg = p2->avg ; p2->avg = t ;

strcpy(temp, p1->num) ; strcpy(p1->num, p2->num) ; strcpy(p2->num, temp) ;

strcpy(temp, p1->name) ; strcpy(p1->name, p2->name) ;strcpy(p2->name, temp) ;

}

}

}

}

void sort(struct student \*\*stu, struct title \*head) //Sort by average grade (exchange pointer field)

{

struct student \*prior1, \*prior2, \*p1, \*p2, \*t ;

char i = 0;

printf("\nSort:\n%-15s%-20s%-10s\n", head->id, head->name, head->avg) ;

p1 = (struct student \*)malloc(sizeof(struct student)) ;

p1->next = \*stu;

(\*stu) = prior1 = p1;

for(p1 = prior1->next; p1->next !=NULL; prior1 = p1, p1 = p1->next ) {

for(p2 = p1->next, prior2 = p1; p2 !=NULL; prior2 = p2, p2 = p2->next) {

if(p1->avg > p2->avg) {

t = p2->next;

prior1->next = p2;

prior2->next = p1;

p2->next = p1->next ;

p1->next = t;

t = p1;

p1 = p2;

p2 = t;

}

}

}

p1 = (\*stu) ;

(\*stu) = (\*stu)->next ;

free(p1) ;

}

int main()

{

char i, j ;

unsigned short m ;

struct student ori, \*head = &ori, \*tail = &ori;

scanf("%hu", &n) ;

for(i = 0; i < n; i ++) { // input record

scanf("%s %s", tail->num, tail->name) ;

scanf("%f %f %f %f", &tail->eng, &tail->cal, &tail->phy, &tail->cpp) ;

getchar();

tail->next = (struct student \*)malloc(sizeof(struct student)) ;

tail = tail->next;

}

tail->next = NULL;

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n", h->id, h->name, h->eng, h->cal, h->phy, h->c) ;

tail = &ori;

while(tail->next) {

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n", tail->num, tail->name, tail->eng, tail->cal, tail ->phy, tail->cpp) ;

tail = tail->next;

}

putchar('\n') ;

scanf("%hu", &m) ;

struct alter top, \*fore = &top, \*aft = &top ;

for(i = 0; i < m; i ++) { // Enter new grades

scanf("%s %s", aft->num, aft->sub) ;

scanf("%f", &aft->score) ;

getchar();

aft->next = (struct alter \*)malloc(sizeof(struct alter)) ;

aft = aft->next;

aft->next = NULL;

}

aft = &top;

tail = &ori;

while(aft) {

tail = &ori;

amend(tail, aft, h) ;

aft = aft->next;

}

printf("Alter:\n%-15s%-20s%-10s%-10s%-10s%-10s\n", h->id, h->name, h->eng, h->cal, h ->phy, h->c) ;

while(tail->next) {

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n", tail->num, tail->name, tail->eng, tail->cal, tail ->phy, tail->cpp) ;

tail = tail->next;

}

tail = &ori;

stat(tail, h) ;

tail = &ori;

sort(tail, h) ;

while(tail->next) {

printf("%-15s%-20s%-10.2f\n", tail->num, tail->name, tail->avg) ;

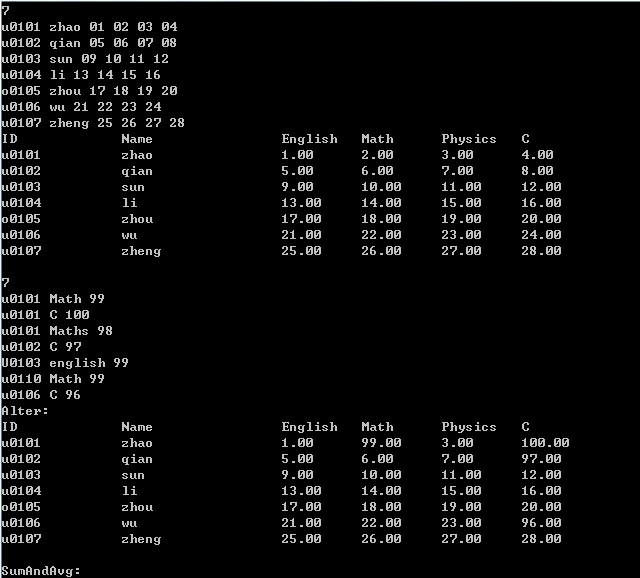
tail = tail->next;

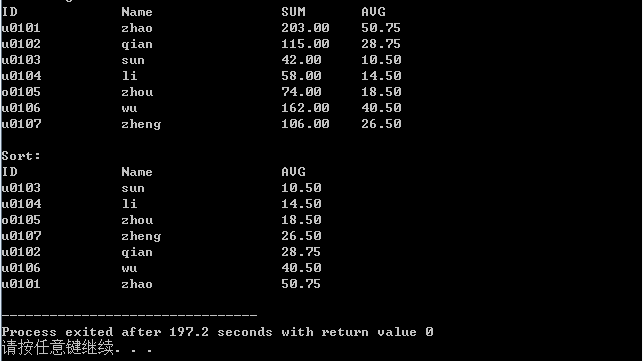
}

}

3 ) Test

( b ) Screenshot of the running results corresponding to the test data





## 7.2 Optional questions

Programming: 3 questions

1. For the program of the programming design question (2), add a function to sort in ascending order according to the average score, and write a function to sort in ascending order by exchanging node data fields. The sorting method can be selected or bubbled. ( Merged with the second question of programming )

2. For the optional question (1), further write the function of sorting in ascending order by exchanging the pointer field of the node.

cooperate with the second question of program design )

3. Rework the programming design question (2) using a doubly linked list.

NULL

## 7.3 Self-made questions

## 7.4 Summary

Personally, I feel that this experiment is simpler than pointers as a whole. One is that the structural union part is easier to learn than pointers, and the other is that the pointers learned earlier have been digested to a considerable extent , and the use of pointers in this part is not so deep .

The previous part of evaluation and verification is the basic content of the structure, but it also has the meaning of infiltrating complex expressions. It is necessary to further deepen the understanding and memory of the priority combination table. Program modification and replacement can be inferred as long as you understand the pointer to itself , these two parts can be said to be not difficult.

The first difficulty of the field structure question is its understanding . The point is that after declaring a field structure, the value stored in the address is certain, the difference is only in the way to access the content in the address, and it should also be remembered that the variable declared first in the field structure corresponds to the low byte, and the variable declared later The corresponding high byte . The diversity of byte access is also the greatest subtlety of the field structure itself. In addition, this question also involves an array of function pointers , which is a big obstacle for me who will have time to practice more in the future, but fortunately it was solved smoothly in the end .

With the rich experience in writing large programs in the past , and the wave of exams has passed, I can finally settle down in the face of this score processing complex. The troublesome problem of mixed input of strings and numbers has been solved in the last experiment. After peeling off the coat of the structure type, what is inside is actually the basic input and output, parameter passing, and array sorting. It should be noted that when exchanging the sorting of data fields, it is only necessary to exchange the parts to be output, but the disadvantage is that the one-to-one correspondence between people and grades is disrupted , making subsequent operations more difficult. Therefore, a safer method is to exchange all data fields , or use a more complicated exchange pointer field.

the second data structure we have come into contact with ( the first is an array ) , but the most extensive application of it in the class is concentrated on the one-way linked list, so if you want to master it more deeply, you still have to try the two-way Use of linked lists and cross-lists. One final thing to note is that a struct is a data type and should not be confused with a class in C ++ .

# 8 file experiments

## 8.1 Required Questions

(1) Program verification of file types: 1 question

With program:

#include <stdio.h>

int main(void)

{

short a=0x253f,b=0x7b7d;

char ch;

FILE \*fp1,\*fp2;

fp1=fopen("d:\\abc1.bin","wb+");

fp2=fopen("d:\\abc2.txt","w+");

fwrite(&a,sizeof(short),1,fp1);

fwrite(&b,sizeof(short),1,fp1);

fprintf(fp2,"%hx %hx",a,b);

rewind(fp1); rewind(fp2);

while((ch = fgetc(fp1)) != EOF)

putchar(ch);

putchar('\n');

while((ch = fgetc(fp2)) != EOF)

putchar(ch);

putchar('\n');

fclose(fp1);

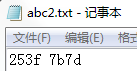
fclose(fp2);

return 0;

}

1. Consider the output of the program, and verify it by running it on a computer.

fp1:253f7b7d; fp2:253f 7b7d; screen :?%}{ ( new line) 253f 7b7d

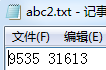
fp2: screen:

1. What is the difference between changing the two sizeof(short) to sizeof( char ) and why?

Screen:

two short low-byte parts are written , namely 3f and 7d (hexadecimal), so when fp1 is output on the screen, only output?}

(3) What is the difference in the result of changing fprintf(fp2,"%hx %hx",a,b) to fprintf(fp2,"% d % d ",a,b) ?

fp2: screen :

The two numbers a and b are converted into decimal and then written into fp2 and printed on the screen

(2) Program modification and replacement: 2 questions

Display the contents of the specified text file on the screen. The format of the command line is:

type filename

1. What kind of logic errors exist in the source program (observe the execution result first)? Modify and debug the program so that it can correctly complete the specified tasks.

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char\* argv[])

{

char ch;

FILE \*fp;

if(argc!=2){

printf("Arguments error!\n");

exit(-1);

}

if((fp=fopen(argv[1],"r"))==NULL){ /\* fp points to filename \*/

printf("Can't open %s file!\n", argv[1]);

exit(-1);

}

while(ch=fgetc(fp)!=EOF) /\* read characters from filename\*/

putchar(ch); /\* write characters to the display\*/

fclose(fp); /\* close filename \*/

return 0;

}

**answer :**

The replaced program looks like this :

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char\* argv[])

{

char ch;

FILE \*fp;

if(argc!=2){

printf("Arguments error!\n");

exit(-1);

}

if((fp=fopen(argv[1],"r"))==NULL){ /\* fp points to filename \*/

printf("Can't open %s file!\n", argv[1]);

exit(-1);

}

while((ch=fgetc(fp))!=EOF) /\* Read characters from filename\*/

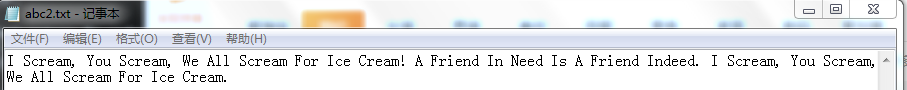
putchar(ch); /\* write characters to the display\*/

fclose(fp); /\* close filename \*/

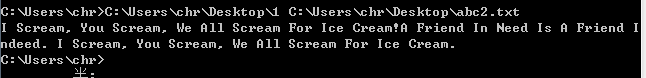
return 0;

}

abc2.txt



cmd



(2) Use input and output redirection freopen to rewrite the main function in the above source program.

**answer :**

The replaced program looks like this :

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char\* argv[])

{

char ch;

FILE \*fp;

if(argc!=2){

printf("Arguments error!\n");

exit(-1);

}

if((fp=freopen(argv[1],"r" ,stdin ))==NULL){ /\* fp points to filename \*/

printf("Can't open %s file!\n", argv[1]);

exit(-1);

}

while((ch=fgetc(fp))!=EOF) /\* Read characters from filename\*/

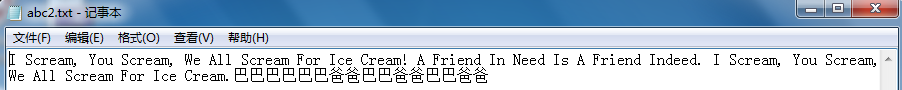
putchar(ch); /\* write characters to the display\*/

fclose(fp); /\* close filename \*/

return 0;

}

abc2.txt



cmd



⑶ Programming: 1 question

Input a line of English sentences from the keyboard, replace the first letter of each word with an uppercase letter, and then output it to a disk file " test" for storage.

**answer:**

1 ) Problem-solving ideas

1. Define the file pointer \*fout, through which the output text file is opened in an update ("a") mode ;

2. Read a line of statements from the standard input device. The feature of the statement with spaces determines that it is more appropriate to use fgets to read in the string , but pay attention to the end of the string with '\0';

3. Use the loop body to judge from the second character , and only need to judge the first letter of the word , that is, if s[i] == ' ' && s[i+1] >= 'a' && s[i+1] <= 'z' , change s[i+1] to uppercase;

4. Separately judge s[0] , if s[0] is a lowercase letter , convert it to uppercase ;

5. Output the modified string in the folder ;

6. Output the modified string on the standard output device ;

7. End \_

2 ) Source program list

#include <stdio.h>

#include <string.h>

int main()

{

char s[100], i ;

FILE \*fout;

fout = fopen("C:\\Users\\chr\\Desktop\\abc2.txt", "a") ;

fgets(s, 100, stdin) ;

s[strlen(s) - 1] = '\0' ;

if(s[0] >= 'a' && s[0] <= 'z') {

s[0] = s[0] - 'a' + 'A' ;

}

for(i = 0; s[i+1] != '\0'; i ++ ) {

if(s[i] == ' ' && s[i+1] >= 'a' && s[i+1] <= 'z') {

s[i+1] = s[i+1] - 'a' + 'A' ;

}

}

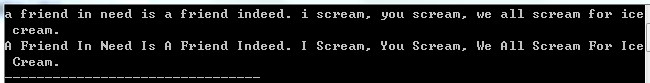
fputs(s, fout) ;

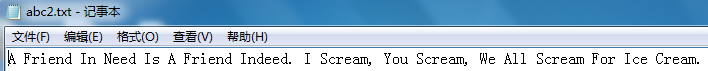
printf("%s", s) ;

}

3 ) Test

Screenshot of the running results corresponding to the test data





## 8.2 Self-made questions

## 8.3 Summary

Since the experiments in this chapter focus on the use of file I/O ports , the content of the programming part is relatively easy to get started , but the difficulty lies in a large number of file-related functions that need to be understood and mastered. Unfortunately, the questions actually only cover the first half of the textbook . For the second half, you still have to go down and find the questions yourself and practice more. Since I didn't fully grasp the command line parameters in the last experiment , I encountered some troubles when using it this time, but finally solved it smoothly . I am more familiar with the understanding and use of command line parameters , the most important point of which is It is the address of the target executable file that occupies the first character string. It is because I did not figure this out that I wasted a lot of time on the command line parameters.

In the previous programming, I simply thought that if the program is easy to understand, you only need to add spaces between words, but last week I saw that C language programming has its own format specification , which is specific to comment style, Code space requirements, long line splitting and blank line requirements. A unified programming style specification can improve the readability of the code , and help the programmer himself or other programmers to thoroughly understand the code in the shortest possible time. Therefore, starting from this assignment , my source code began to gradually become closer to the specification. However, since this part of the work was carried out uniformly after the program was running normally, it did not have a great impact on the efficiency of completing the task.

## references

[1] Cao Jichang, Lu Ping, Li Kai. C Language Programming, Beijing: Science Press , 2013

[2] Li Kai, Lu Ping, Cao Jichang. C language experiment and curriculum design, Beijing: Science Press , 2011