

华中科技大学

课程实验报告

课程名称： C 语言程序设计

专业班级： CS1601

学 号： U201614531

姓 名： 刘本嵩

指导教师： 甘早斌

报告日期： 2017 年 6 月 4 日

计算机科学与技术学院

目 录

1	表达式和标准输入输出实验.....	4
1.1	实验目的.....	4
1.2	实验内容.....	4
1.3	自设题.....	5
1.4	实验小结.....	8
2	流程控制实验.....	9
2.1	实验目的.....	9
2.2	实验内容.....	9
2.3	自设题.....	10
2.4	实验小结.....	13
3	函数与程序结构实验.....	14
3.1	实验目的.....	14
3.2	实验内容.....	14
3.3	自设题.....	20
3.4	实验小结.....	22
4	编译预处理实验.....	23
4.1	实验目的.....	23
4.2	实验内容.....	23
4.3	自设题.....	27
4.4	实验小结.....	29

5	数组实验.....	30
5.1	实验目的.....	30
5.2	实验内容.....	30
5.3	自设题.....	35
5.4	实验小结.....	39
6	指针实验.....	40
6.1	实验目的.....	40
6.2	实验内容.....	40
6.3	自设题.....	44
6.4	实验小结.....	48
7	结构与联合实验.....	49
7.1	实验目的.....	49
7.2	实验内容.....	49
7.3	自设题.....	52
7.4	实验小结.....	57
8	文件实验.....	58
8.1	实验目的.....	58
8.2	实验内容.....	58
8.3	自设题.....	60
8.4	实验小结.....	60
	参考文献.....	61

实验 1 表达式和标准输入与输出实验

1.1 实验目的

(1) 熟练掌握各种运算符的运算功能，操作数的类型，运算结果的类型及运算过程中的类型转换，重点是 C 语言特有的运算符，例如位运算符，问号运算符，逗号运算符等；熟记运算符的优先级和结合性；

(2) 掌握 `getchar`, `putchar`, `scanf` 和 `printf` 函数的用法。

(3) 掌握简单 C 程序（顺序结构程序）的编写方法。

1.2 实验内容

1、源程序改错题

下面给出了一个简单 C 语言程序例程，用来完成以下工作：

1. 输入华氏温度 `f`，将它转换成摄氏温度 `c` 后输出；
2. 输入圆的半径值 `r`，计算并输出圆的面积 `s`；
3. 输入短整数 `k`、`p`，将 `k` 的高字节作为结果的低字节，`p` 的高字节作为结果的高字节，拼成一个新的整数，然后输出；

在这个例子程序中存在若干语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。

```
1  #include<stdio.h>
2  #define PI 3.14159;
3  intmain( void )
4  {
5      int f;
6      short p, k;
7      double c, r, s;
8      /* for task 1 */
9      printf("Input  Fahrenheit:" );
10     scanf("%d", f);
11     c = 5/9*(f-32);
```

```

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

13  /* for task 2 */
14  printf("input the radius r:");
15  scanf("%f", &r);
16  s = PI * r * r;
17  printf("\nThe acreage is %.2f\n\n",&s);
18  /* for task 3 */
19  printf("input hex int k, p :");
20  scanf("%x %x", &k, &p );
21  newint = (p&0xff00)|(k&0xff00)<<8;
22  printf("new int = %x\n\n",newint);
23  return 0;
}

```

解答：

（1）错误修改：

line2 Dangerous alias.

line3 undefined identifier 'intmain'. 缺少空格

line5 f 为浮点数

line6 sizeof(short) is machine-dependent, 改为 uint16_t

line11-13 17 继承上述错误，一并修改为浮点运算。

line19 printf 参数 2 期望浮点数，实际传入指针。

line21 逻辑错误，重写

（2）错误修改后运行结果：

```

recolic@RECOLICPC ~/c/h/1/zzsrc> uname -a
Linux RECOLICPC 4.10.1-1-ARCH #1 SMP PREEMPT Sun Feb 26 21:08:53 UTC 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/1/zzsrc> gcc --version
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权声明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
recolic@RECOLICPC ~/c/h/1/zzsrc> cat 1.tst
100
2
dd ff

recolic@RECOLICPC ~/c/h/1/zzsrc> gcc -std=c11 -o exel 1.c; and ./exel < 1.tst
Input Fahrenheit:
 100.000000 (F) = 37.78 (C)

input the radius r:
The acreage is 12.57

input hex int k, p :new int = 0
recolic@RECOLICPC ~/c/h/1/zzsrc> 

```

```

#include<stdio.h>
#include<stdint.h>
#define PI 3.14159
int main()
{
    double f;
    uint16_t p, k;
    double c, r, s;

    /* for task 1 */
    printf("Input  Fahrenheit:" );
    scanf("%lf", &f );
    c = 5.0/9.0*(f-32.0);
    printf( " \n %lf (F) = %.2f (C)\n\n ", f, c );

    /* for task 2 */
    printf("input the radius r:");
    scanf("%lf", &r);
    s = PI * r * r;
    printf("\nThe acreage is %.2f\n\n",s);

    /* for task 3 */
    printf("input hex int k, p :");
    scanf("%x %x", &k, &p );
    long long newint = ((p >> 8) << 8) + (k >> 8);
    printf("new int = %x\n\n",newint);
}

```

2、源程序修改替换题

下面的程序利用常用的中间变量法实现两数交换，请改用不用第三个变量的交换法实现。

解答：替换后的程序如下所示：

```

void swapIntByP(int *swa, int *swb)
{
    #define __a_66 (*swa)
    #define __b_66 (*swb)
    __a_66 = __a_66 ^ __b_66;
    __b_66 = __b_66 ^ __a_66;
    __a_66 = __a_66 ^ __b_66;
    return;
}
#include<stdio.h>
int main()
{
    int a, b;
    printf("Input two integers:");
    scanf("%d %d",&a,&b);
    swapIntByP(&a,&b);
    printf("\na=%d,b=%d",a,b);
}

```

```

recolic@RECOLICPC ~/c/h/1/zzsrc> uname -a
Linux RECOLICPC 4.10.1-1-ARCH #1 SMP PREEMPT Sun Feb 26 21:08:53 UTC 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/1/zzsrc> gcc --version
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权声明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
recolic@RECOLICPC ~/c/h/1/zzsrc> cat 2.tst
123 666

recolic@RECOLICPC ~/c/h/1/zzsrc> gcc -std=c11 -o exe2 1-2.c; and ./exe2 < 2.tst
Input two integers:
a=666,b=123~
recolic@RECOLICPC ~/c/h/1/zzsrc> █

```

3、编程设计题

上机调试运行以下程序：

(1) 编写一个程序，输入字符 c ，如果 c 是大写字母，则将 c 转换成对应的小写，否则 c 的值不变，最后输出 c 。

//I'm sorry but have to add a '\n' character while print char "c". If not, stdout under linux causes problem while dealing stdout.

```

#include <stdio.h>
int main()
{
    char iCh = '\0';
    scanf("%c", &iCh);
    if(iCh <= 'Z' && iCh >= 'A')
        iCh -= ('A' - 'a');
    printf("%c\n", iCh);
    return 0;
}

```

(2) 编写一个程序，输入无符号短整数 x ， m ， n ($0 \leq m \leq 15$, $1 \leq n \leq 16-m$)，取出 x 从第 m 位开始向左的 n 位 (m 从右至左编号为 $0 \sim 15$)，并使其向左端 (第 15 位) 靠齐。

```

#include <stdio.h>
int main()
{
    unsigned short x, m, n;
    scanf("%hu %hu %hu", &x, &m, &n);
    unsigned short mask1, mask2;
    if(m < 1)
        mask1 = ~0;
    else
        mask1 = (~0) << m;
    mask2 = (~0) >> (16-m-n);
    x = x & mask1 & mask2;
    x = x << (16-m-n);
    printf("%hu\n", x);
    return 0;
}

```

```
}
```

(3) IP 地址通常是 4 个用句点分隔的小整数，如 32.55.1.102。这些地址在机器中用无符号长整形表示。编写一个程序，以机器存储的形式读入一个 32 位的互联网 IP 地址，对其译码，然后用常见的句点分隔的 4 部分的形式输出。

```
#include <stdio.h>
int main()
{
    int N;
    scanf("%d", &N);
    for(int cter=1; cter<=N; ++cter)
    {
        unsigned int def_msk;
        scanf("%u", &def_msk);
        unsigned mask1, mask2, mask3, mask4;
        mask1=(~0)<<24;
        mask2=mask1>>8;
        mask3=mask1>>16;
        mask4=mask1>>24;
        unsigned nodeA, nodeB, nodeC, nodeD;
        nodeA = (def_msk&mask1)>>24;
        nodeB = (def_msk&mask2)>>16;
        nodeC = (def_msk&mask3)>>8;
        nodeD = def_msk&mask4;
        printf("%d.%d.%d.%d\n", nodeD, nodeC, nodeB, nodeA);
    }
    return 0;
}
```

1.3 自设题

(1) 自设实验题目：实型数的浮点表示误差的简单验证

(2) 实验目的：通过设计实验程序，理解实数的浮点数表示上的误差，

浮点数不宜做等值比较。

(3) 实验程序：

```
#include <iostream>
using namespace std;
int main()
{
    float fTst = 1.0 / 3.0;
    double dTst = 10.0 / 30.0;
    cout << boolalpha << (static_cast<double>(fTst) == dTst) << endl;
    return 0;
}
//////////
g++ (GCC) 6.3.1 20170306
Copyright (C) 2016 Free Software Foundation, Inc.
```


This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

(4) 实验用例：用例写入代码内

(5) 实验结论：浮点数做等值比较会产生与常识不一致的结论。

实验 2 流程控制实验

2.1、实验目的

(1) 掌握复合语句、if 语句、switch 语句的使用,熟练掌握 for、while、do-while 三种基本的循环控制语句的使用,掌握重复循环技术,了解转移语句与标号语句。

(2) 练习循环结构 for、while、do-while 语句的使用。

(3) 练习转移语句和标号语句的使用。

(4) 使用 Turbo C 2.0 集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

2.1、实验内容

1. 源程序改错题

下面是计算 $s=n!$ 的源程序,在这个源程序中存在若干语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改,使之能够正确完成指定任务。例如,

$8! = 40320$ 。

```
#include <stdio.h>
void main(void)
{
```

```
int i,n,s=1;
printf("Please enter n:");
scanf("%d",&n);
for(i=1,i<=n,i++)
    s=s*i;
printf("%d! = %d",n,s);
}
```

解答:

(1) 错误修改:

1) 21.c:7:19: Error: expected ';' before ')' token

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

2) 21.c:6:19: logic_rror: scanf 参数 2 期望 int* 实际传入 int

```
>>> scanf("%d",&n);
```

(2) 错误修改后运行结果:

(给一个运行截图简单说明修改方案的正确性)



```
recolic@RECOLICPC ~/c/h/s/17c> gcc --version
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权声明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
recolic@RECOLICPC ~/c/h/s/17c> cat 21.c
#include <stdio.h>
void main(void)
{
    int i,n,s=1;
    printf("Please enter n:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        s=s*i;
    printf("%d! = %d",n,s);
}
recolic@RECOLICPC ~/c/h/s/17c> gcc -std=c11 21.c -o 21_ and echo '8' > 21.in_ and ./21_ < 21.in
Please enter n:8! = 40320
```

2. 源程序修改替换题

(1) 修改第 1 题，分别用 while 和 do-while 语句替换 for 语句。

```
#include <stdio.h>
void main(void)
{
    int i,n,s=1;
    printf("Please enter n:");
    scanf("%d",&n);
    i=1
    while(i<=n)
    {
        s*=i;
```

```

        ++i;
    }
    printf("%d! = %d",n,s);
}
#include <stdio.h>
void main(void)
{
    int i,n,s=1;
    printf("Please enter n:");
    scanf("%d",&n);
    i=1
    do{
        If(i==1)continue;
        s*=i;
        ++i;
    }while(i<=n);
    printf("%d! = %d",n,s);
}

```

(2) 修改第 1 题, 输入改为“整数 S”, 输出改为“满足 $n! \geq S$ 的最小整数 n”。

例如输入整数 40310, 输出结果为 n=8。

```

#include <stdio.h>

unsigned long long int next()
{
    static int i = 0;
    ++i;
    static unsigned long long int current = 1;
    current *= i;
    // printf(">%lld", current);
    return current;
}

int getRes(unsigned long long int inpInt)
{
    size_t cter = 1;
    for(; next() < inpInt; ++cter);
    return cter;
}

int main()
{
    int i,n,s=1;
    printf("Please enter n:");
    scanf("%d",&n);
    int res = getRes((unsigned long long int)n);
    printf("%d\n",res);
    return 0;
}

```

3. 编程设计题

(1) 打印如下杨辉三角形。

```

1
1 1
1 2 1
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

```

每个数据值可以由组合 C_i^j 计算（表示第 i 行第 j 列位置的值），而 C_i^j 的计算如下：

$$C_i^0 = 1 \quad (i=0,1,2,\dots)$$

$$C_i^j = C_i^{j-1} * (i-j+1) / j \quad (j=0,1,2,3,\dots,i)$$

本程序中为了打印出金字塔效果，要注意空格的数目。一位数之间是 3 个空格，两位数之间有 2 个空格，3 位数之间只有一个空格，程序编制过程中要注意区分。

```
#include <stdio.h>
int Yang(int j,int l,int n);
int JC(int x);
int main()
{
    int i,l,j,n;
    for(scanf("%d",&n);n!=0;scanf("%d",&n)){
        for(l=0;l<=n-1;l++){
            for(i=1,j=1;j<=l+1;i++){
                {
                    if(i<=2*(n-1-l))
                        putchar(' ');
                    else
                        {
                            Yang(j,l,n);
                            j++;
                        }
                }
            }
        }
    }
}
```

```

        printf("\n");
    }
    printf("\n");
}
return 0;
}
int Yang(int j,int l,int n)
{
    int m=JC(l)/(JC(j-1)*JC(l+1-j));
    if(j==1)
        printf("%d",m);
    else
        printf("%4d",m);
}
int JC(int x)
{
    int k,s=1;
    for(k=1;k<=x;k++)
        s*=k;
    return s;
}

```

编写一个程序,将用户输入的任意正整数逆转,例如,输入 1234,输出 4321。

```

#include<stdio.h>
long t10(int x);
int main()
{
    long m;
    int s[100],n=1,i,j;
    scanf("%ld",&m);
    while(m){
        i=m;
        while(i>=10){
            i/=10;
            n++;
        }
        long S=0;
        for(j=1;j<=n;j++){
            s[j]=m%10;
            m/=10;
            S=S+s[j]*t10(n-j);
        }
        printf("%ld\n",S);
        scanf("%ld",&m);
        n=1;
    }
    return 0;
}
long t10(int x)
{
    int k;
    long y=1;
    for(k=1;k<=x;k++){
        y*=10;
    }
    return y;
}

```

3 函数与程序结构实验

3.1 实验目的

- (1) 熟悉和掌握函数的定义、声明；函数调用与参数传递方法；以及函数返回值类型的定义和返回值使用。
- (2) 熟悉和掌握不同存储类型变量的使用。
- (3) 熟悉多文件编译技术。

3.2、实验内容

1. 源程序改错题

下面是计算 $s=1!+2!+3!+\dots+n!$ 的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。

```
#include "stdio.h"
Int main(void)
{
    int k;
    for(k=1;k<6;k++)
        printf("k=%d\tthe sum is %ld\n",k,sum_fac(k));
}
long sum_fac(int n)
{
    Static long s=0;
    int i;
    long fac=1;
    for(i=1;i<=n;i++)
        fac*=i;
    s+=fac;
    return s;
```

```
}
```

解答:

(1) 错误修改:

31.c:6:44: 警告: 隐式声明函数'sum_fac'

2+ long sum_fac(int n);

3-void+int 会造成 linuxshell 返回随机值, 不符合 POSIX C Standard

10+static 逻辑错误

12+init=1 逻辑错误

(2) 错误修改后运行结果:

```
recolic@RECOLICPC ~/c/h/s/17c [19]> gcc --version
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件: 请参看源代码的版权说明。本软件没有任何担保;
包括没有适销性和某一专用目的下的适用性担保。不同存储类型变量的使用。
recolic@RECOLICPC ~/c/h/s/17c> uname -a
Linux RECOLICPC 4.10.8-1-ARCH #1 SMP PREEMPT Fri Mar 31 16:50:19 CEST 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/s/17c> cat 31.c
#include "stdio.h"
long sum_fac(int n);
void main(void)
{
    int k;
    for(k=1;k<6;k++)
        printf("k=%d\tthe sum is %ld\n",k,sum_fac(k));
}
long sum_fac(int n)
{
    static long s=0;
    int i;
    long fac=1;
    for(i=1;i<=n;i++)
        fac*=i;
    s+=fac;
    return s;
}
recolic@RECOLICPC ~/c/h/s/17c> gcc -std=c11 31.c -o 31; ./31
k=1 the sum is 1
k=2 the sum is 3
k=3 the sum is 9
k=4 the sum is 33
k=5 the sum is 153
recolic@RECOLICPC ~/c/h/s/17c [19]>
```

2. 源程序修改替换题

(1) 修改第 1 题中 sum_fac 函数, 使其计算量最小。

```
#include "stdio.h"
#include <stdint.h>
uint64_t dpBuffer[64];
uint64_t getf(size_t n)
{
```

```

        if(dpBuffer[n] == 0)
            dpBuffer[n] = getf(n-1)*n;
        return dpBuffer[n];
    }
    uint64_t sum_fac(int n)
    {
        uint64_t s = 0;
        for(size_t cter = 1; cter < n+1; ++cter)
            s += getf(cter);
        return s;
    }
    void main(void)
    {
        for(size_t cter = 0; cter < 64; ++cter)
            dpBuffer[cter] = 0;
        dpBuffer[0] = 1;
        dpBuffer[1] = 1;
        dpBuffer[2] = 2;

        int k;
        for(k=1;k<6;k++)
            printf("k=%d\tthe sum is %ld\n",k,sum_fac(k));
    }

```



```

recolic@RECOLICPC ~/c/h/s/17c> gcc --version
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权声明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
recolic@RECOLICPC ~/c/h/s/17c> uname -a
Linux RECOLICPC 4.10.8-1-ARCH #1 SMP PREEMPT Fri Mar 31 16:50:19 CEST 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/s/17c> cat 32.c
#include "stdio.h"
#include <stdint.h>
uint64_t dpBuffer[64];
uint64_t getf(size_t n)
{
    if(dpBuffer[n] == 0)
    {
        dpBuffer[n] = getf(n-1)*n;
        return dpBuffer[n];
    }
}
uint64_t sum_fac(int n)
{
    uint64_t s = 0;
    for(size_t cter = 1; cter < n+1; ++cter)
        s += getf(cter);
    return s;
}
void main(void)
{
    for(size_t cter = 0; cter < 64; ++cter)
        dpBuffer[cter] = 0;
    dpBuffer[0] = 1;
    dpBuffer[1] = 1;
    dpBuffer[2] = 2;

    int k;
    for(k=1; k<6; k++)
        printf("k=%d\tthe sum is %ld\n", k, sum_fac(k));
}
recolic@RECOLICPC ~/c/h/s/17c> gcc -std=c11 32.c -o 32; ./32
k=1 the sum is 1
k=2 the sum is 3
k=3 the sum is 9
k=4 the sum is 33
k=5 the sum is 153
recolic@RECOLICPC ~/c/h/s/17c [19]>

```

(2) 修改第 1 题中 sum_fac 函数，计算 $s = 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$ 。

```

#include "stdio.h"
#include <stdint.h>
uint64_t dpBuffer[64];
uint64_t getf(size_t n)
{
    if(dpBuffer[n] == 0)
        dpBuffer[n] = getf(n-1)*n;
    return dpBuffer[n];
}
uint64_t sum_fac(int n)
{
    long double s = 0;
    for(size_t cter = 1; cter < n+1; ++cter)
        s += getf(cter);
    return s;
}
void main(void)
{
    for(size_t cter = 0; cter < 64; ++cter)

```

```

        dpBuffer[cter] = 0;
dpBuffer[0] = 1;
dpBuffer[1] = 1;
dpBuffer[2] = 2;

int k;
for(k=1;k<6;k++)
    printf("k=%d\tthe sum is %ld\n",k,sum_fac(k));
}

```

```

recolic@RECOLICPC ~/c/h/s/17c [24]> gcc --version; and uname -a
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权说明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
Linux RECOLICPC 4.10.8-1-ARCH #1 SMP PREEMPT Fri Mar 31 16:50:19 CEST 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/s/17c> cat 33.c
#include "stdio.h"
#include <stdint.h>
uint64_t dpBuffer[64];
uint64_t getf(size_t n)
{
    if(dpBuffer[n] == 0)
        dpBuffer[n] = getf(n-1)*n;
    return dpBuffer[n];
}
long double sum_fac(int n)
{
    long double s = 0;
    for(size_t cter = 1; cter < n+1; ++cter)
        s += (long double)1/getf(cter);
    return s;
}
void main(void)
{
    for(size_t cter = 0; cter < 64; ++cter)
        dpBuffer[cter] = 0;
    dpBuffer[0] = 1;
    dpBuffer[1] = 1;
    dpBuffer[2] = 2;

    int k;
    for(k=1;k<6;k++)
        printf("k=%d\tthe sum is %Lf\n",k,sum_fac(k));
}
recolic@RECOLICPC ~/c/h/s/17c> gcc -std=c11 33.c -o 33; ./33
k=1    the sum is 1.000000
k=2    the sum is 1.500000
k=3    the sum is 1.666667
k=4    the sum is 1.708333
k=5    the sum is 1.716667
recolic@RECOLICPC ~/c/h/s/17c [24]>

```

From the printf manpage:

L (ell) A following integer conversion corresponds to a following n conversion corresponds to a pointer conversion corresponds to a wint_t argument, or a pointer to wchar_t argument.

and

L A following a, A, e, E, f, F, g, or G conversion corresponds to a long double conversion. The conversion allows %LF, but SUSv2 does not.)

So, you want %Le, not %le

Edit: Some further investigation seems to indicate that stuff like printf) - which maps long double to double: native long double with a runtime that does not seem to be supported.

I found linux.die.net/man/3/printf, but %Lf or %Le also require the -std=c11 flag.

Which compiler are you using with Netbeans on Windows?

latest gcc and MinGW – gameboy Nov 3 '10 at 17:13

Yes -- for long double, you need to use %Lf (i.e. %Le is for double).

3. 跟踪调试题

计算 fabonacci 数列前 n 项和的程序如下：

其中，long sum=0,*p=∑声明 p 为长整型指针并用&sum 取出 sum

的地址对 p 初始化。*p 表示引用 p 所指的变量（*p 即 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);
    }
}

```

单步执行程序，观察 p,i,sum,n 值。

(1) 刚执行完 scanf("%d",&k);语句，p,i 值是多少？

(2) 从 fabonacci 函数返回后光条停留在哪个语句上？

(3) 进入 fabonacci 函数，watch 窗口显示的是什么？

(4) 当 i=3，从调用 fabonacci 函数到返回，n 值如何变化？

34.c: 在函数'main'中:

34.c:8:14: 警告：隐式声明函数'fabonacci' [-Wimplicit-function-declaration]
 sum+=fabonacci(i);
 ^~~~~~

34.c: 在文件作用域:

34.c:12:6: 错误：与'fabonacci'类型冲突
 long fabonacci(int n)
 ^~~~~~

34.c:8:14: 附注：'fabonacci'的上一个隐式声明在此
 sum+=fabonacci(i);
 ^~~~~~

已修改为：

```

#include <stdio.h>
long fabonacci(int n)
{
    if(n==1 || n==2)
        return 1;
    else
        return fabonacci(n-1)+fabonacci(n-2);
}
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);
    }
}
(1)

```

```

(gdb) print p
$1 = (long *) 0x7fffffffe090
(gdb) print i
$2 = 0
(2) 没有光条出现。你指的是 sum+=fabonacci(i); ? 这和 IDE 的具体实现有关
(3) 所有局部变量如下 :
(gdb) break 3
Breakpoint 1 at 0x400572: file 34.c, line 3.
(gdb) run
Starting program: /home/recolic/cpp/homework/src/17c/34
5

Breakpoint 1, fabonacci (n=1) at 34.c:4
4      if(n==1 || n==2)
(gdb) print n
$1 = 1
(4) 11 行即为 f(3)调用处 , 执行情况如下 :
(gdb) break 11
Breakpoint 1 at 0x4004f0: file 34-2.c, line 11.
(gdb) run
Starting program: /home/recolic/cpp/homework/src/17c/34-2

Breakpoint 1, main () at 34-2.c:11
11      fabonacci(3);
(gdb) step
fabonacci (n=3) at 34-2.c:4
4      if(n==1 || n==2)
(gdb) s
7      return fabonacci(n-1)+fabonacci(n-2);
(gdb) s
fabonacci (n=2) at 34-2.c:4
4      if(n==1 || n==2)
(gdb) s
5      return 1;
(gdb) s
8  }
(gdb) s
fabonacci (n=1) at 34-2.c:4
4      if(n==1 || n==2)
(gdb) s
5      return 1;
(gdb) s
8  }

```

4. 编程设计题

(1) 编程让用户输入两个整数, 计算两个数的最大公约数并且输出之(要求用递归函数实现求最大公约数)。同时以单步方式执行该程序, 观察递归过程。

```

#include<stdio.h>
int factorial(int n, int m);
int main(void) {
    int n, m;

```

```

        for (;;) {
            scanf("%d%d", &n, &m);
            if (n == 0 || m == 0) break;
            else {
                printf("%d\n", factorial(n, m));
            }
        }
    }
}
int factorial(int n, int m) {
    if (n < m) {
        m = m%n;
        if (m == 0) return n;
        else factorial(n, m);
    }
    else {
        n = n%m;
        if (n == 0) return m;
        else factorial(m, n);
    }
}
}

```

(2) 编程验证歌德巴赫猜想：一个大于等于 4 的偶数都是两个素数之和。

编写一个程序证明对于在符号常量 BEGIN 和 END 之间的偶数这一猜测成

立。例如，如果 BEGIN 为 10，END 为 20，程序的输出应为：

GOLDBACH'S CONJECTURE:

Every even number $n \geq 4$ is the sum of two primes.

10=3+7

12=5+7

.....

20=3+17

```

#include<stdio.h>
int Primes(int x) /*判断素数函数*/
{
    int i,j;
    if(x==2) return 1;
    if(!(x%2)) return 0;
    j=x/2;
    for(i=3;i<=j;i+=2)
        if(!(x%i)) return 0;
    return 1;
}
int main()
{
    int num,i,m,n;
    while(scanf("%d%d",&m,&n)&&m!=0&&n!=0)
    {
        if(m%2!=0)

```

```

        m=m+1;
    for(num=m;num<=n;num+=2)
    {
        for(i=2;i<=num/2;i++)
        {
            if(Primes(i)&&Primes(num-i))
            {
                printf("%d=%d+%d\n",num,i,num-i);
                break;
            }
        }
        printf("\n");
    }
    return 0;
}

```

5. 选做 s 题

1、设 file1.c 如下：

```

#include <stdio.h>
int x,y; /* 外部变量的定义性说明 */
char ch; /* 外部变量的定义性说明 */
void main(void)
{
    x=10;
    y=20;
    ch=getchar();
    printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);
    func1();
}

```

file2.c 如下：

```

extern int x,y; /* 外部变量的引用性说明 */
extern char ch; /* 外部变量的引用性说明 */
void func1(void)
{
    x++;
    y++;
    ch++;
    printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);
}

```

试用 TCC 进行多文件编译和链接。然后在 DOS 环境下运行生成的可执行文件。

没有 Dos，只有 Cmake。

```

recolic@RECOLICPC ~/c/h/s/1/5 [27]> make clean
rm *.o 5
recolic@RECOLICPC ~/c/h/s/1/5> cat makefile
def: 5-1.o 5-2.o
    cc -o 5 5-1.o 5-2.o
clean:

```

```
rm *.o 5
recolic@RECOLICPC ~/c/h/s/1/5> make
cc      -c -o 5-1.o 5-1.c
cc      -c -o 5-2.o 5-2.c
cc -o 5 5-1.o 5-2.o
recolic@RECOLICPC ~/c/h/s/1/5> ./5
testString
in file1 x=10,y=20,ch is t
in file2 x=11,y=21,ch is u
```

实验 4 编译预处理实验

一、实验目的

掌握文件包含、宏定义、条件编译、assert 宏的使用；

练习带参数的宏定义、条件编译的使用；

练习 assert 宏的使用；

使用 Turbo C 2.0 集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

二、实验题目及要求

1. 源程序改错题

下面是用宏来计算平方差、交换两数的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。

```
#include "stdio.h"
#define SUM a+b
#define DIF a-b
#define SWAP(a,b) ({__typeof__(a) __a=a; __typeof__(b) __b=b;a=__b;b=__a;})
void main
{
    int b, t;
    printf("Input two integers a, b:");
    scanf("%d,%d", &a,&b);
    printf("\nSUM=%d\n the difference between square of a and square of b
is:%d",SUM, SUM*DIF);
```

```

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

```

解答:

(1) 错误修改:

1) 41.c:6:1: 错误: expected '=', ',', ';', 'asm' or '__attribute__' before '{' token

```
int main()
```

2) 41.c:9:21: 错误: 'a'未声明(在此函数内第一次使用)

```
int a,b,t;
```

3) 41.c:12:5: 警告: 隐式声明函数'Printf' [-Wimplicit-function-declaration]

```
printf("\nNow a=%d,b=%d\n",a,b);
```

4) 41.c:4:0 错误 逻辑错误

//此改法仅限 GCC

```

#define SWAP(a,b) ({__typeof__(a) __a=a; __typeof__(b)
__b=b;a=__b;b=__a;})

```

(2) 错误修改后运行结果:

```

recolic@RECOLICPC ~/c/h/s/17c> cat 41.c
#include "stdio.h"
#define SUM a+b
#define DIF a-b
#define SWAP(a,b) ({__typeof__(a) __a=a; __typeof__(b) __b=b;a=__b;b=__a;})
int main()
{
    int a, b, t;
    printf("Input two integers a, b:");
    scanf("%d,%d", &a,&b);
    printf("\nSUM=%d\n the difference between square of a and square of b is:%d",SUM, SUM*DIF);
    SWAP(a,b);
    printf("\nNow a=%d,b=%d\n",a,b);
}
recolic@RECOLICPC ~/c/h/s/17c> gcc --version; and uname -a
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权说明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
Linux RECOLICPC 4.10.9-1-ARCH #1 SMP PREEMPT Sat Apr 8 12:39:59 CEST 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/s/17c> gcc -std=c11 41.c -o 41; and ./41
Input two integers a, b:1,2
SUM=3
the difference between square of a and square of b is:1
Now a=2,b=1
recolic@RECOLICPC ~/c/h/s/17c>

```


2. 源程序修改替换题

下面是用函数实现求三个数中最大数、计算两数之和的程序，在这个源程序中存在若干语法和逻辑错误。

要求：1) 对这个例子程序进行调试修改，使之能够正确完成指定任务；

2) 用带参数的宏替换函数 `max`，来实现求最大数的功能。

```
void 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));
}

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;
}
```

解答：

xxxxxxx, xxxxxxx, 替换后的程序如下所示：

```
#define __RIMPL_PASTE__(A,B) A##B
#define MAX(A,B) __MAX_IMPL__(A,B,__COUNTER__)
#define __MAX_IMPL__(A,B,L) ({ __typeof__(A) __RIMPL_PASTE__(_a,L) = (A); \
                                __typeof__(B) __RIMPL_PASTE__(_b,L) = \
                                (B); \
                                (__RIMPL_PASTE__(_a,L)>__RIMPL_PASTE__(_b,L)) \
                                ? __RIMPL_PASTE__(_a,L) : __RIMPL_PASTE__(_b,L); \
                                })

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

```

return MAX(x,MAX(y,z));
}

float sum(float x, float y)
{
    return x+y;
}

int main()
{
    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));
}

```

```

recolic@RECOLICPC ~/c/h/s/17c> cat 42.c
#define __RIMPL_PASTE__(A,B) A##Bxxxxxxxxxxxxxxxx, 替换后的程序如下所示:
#define MAX(A,B) __MAX_IMPL__(A,B,__COUNTER__)
#define __MAX_IMPL__(A,B,L) ({ __typeof__(A) __RIMPL_PASTE__(_a,L) = (A); \
                                __typeof__(B) __RIMPL_PASTE__(_b,L) = (B); \
                                __RIMPL_PASTE__(_a,L) > __RIMPL_PASTE__(_b,L) ? \
                                __RIMPL_PASTE__(_a,L) : __RIMPL_PASTE__(_b,L); \
                                })
#include <stdio.h>

int max(int x, int y, int z)
{
    return MAX(x,MAX(y,z));
}

float sum(float x, float y)
{
    return x+y;
}

int main()
{
    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. 跟踪调试题
recolic@RECOLICPC ~/c/h/s/17c> gcc -std=c11 42.c -o 42 -lm
Enter three integers:12,23,34
the maximum of them is 34
Enter two floating point numbers:1.22,2.33
the sum of them is 3.550000

```

3. 跟踪调试题

下面程序利用 R 计算圆的面积 s，以及面积 s 的整数部分。

```

#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) 修改程序，使程序编译通过且能运行；

```

#include <stdio.h>
#include <assert.h>
#define R r
int integer_fraction(float x);
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
}

int integer_fraction(float x)
{
int i=x;
return i;
}

```

2) 单步执行。进入函数 `decimal_fraction` 时 watch 窗口中 x 为何值？在返回 main 时, watch 窗口中 i 为何值？

x 与输入有关。详情见 gdb 输出

```
(gdb) break 22
Breakpoint 1 at 0x40069b: file 43.c, line 22.
(gdb) run
Starting program: /home/recolic/cpp/homework/src/17c/a.out
input a number: 2
area of round is: 12.566360
```

```
Breakpoint 1, integer_fraction (x=12.5663605) at 43.c:23
23   int i=x;
(gdb) print x
$1 = 12.5663605
(gdb) print i
$2 = 0
(gdb) step
24   return i;
(gdb) print i
$3 = 12
```

3) 排除错误, 使程序能正确输出面积 s 值的整数部分, 不会输出错误信息

assertion failed。

已经改了。

```
recolic@RECOLICPC ~/c/h/s/17c> gcc --version; and uname -a
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件; 请参看源代码的版权声明。本软件没有任何担保;
包括没有适销性和某一专用目的下的适用性担保。
Linux RECOLICPC 4.10.9-1-ARCH #1 SMP PREEMPT Sat Apr 8 12:39:59 CEST 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/s/17c> cat 43.c
#include <stdio.h>
#include <assert.h>
#define R r
int integer_fraction(float x);
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
}

int integer_fraction(float x)
{
    int i=x;
    return i;
}

recolic@RECOLICPC ~/c/h/s/17c> gcc -std=c11 43.c -o 43; and ./43
input a number: 10
area of round is: 314.158997
the integer fraction of area is 314
```

4. 编程设计题

(1) 三角形的面积是 $area = \sqrt{s(s-a)(s-b)(s-c)}$, 其中 $s = (a+b+c)/2$,
a,b,c 为三角形的三边,定义两个带参数的宏,一个用来求 s,另一个用来求 area。

编写程序,用带参数的宏来计算三角形的面积。

```
#include<stdio.h>
#include<math.h>
#define s(a,b,c) ((a+b+c)/2)
#define area(s,a,b,c) (sqrt(s*(s-a)*(s-b)*(s-c)))
int main ()
{
    int a,b,c,s;
    double area;
    while(scanf("%d %d %d",&a,&b,&c)!=EOF)
    {
        s=s(a,b,c);
        area=area(s,a,b,c);
        printf("%d %lf\n",s,area);
    }
    return 0;
}
```

(2) 用条件编译方法来编写程序。输入一行电报文字,可以任选两种输出:
一为原文输出;二为变换字母的大小写(如小写‘a’变成大写‘A’,大写‘D’
变成小写‘d’),其他字符不变。用#define 命令控制是否变换字母的大小写。

例如, #define CHANGE 1 则输出变换后的文字,若#define CHANGE 0 则原文
输出。

```
#include <stdio.h>
#define CHANGE (c%2)
int main(void)
{
    char c,s;
    int flag1=0,flag2=1;
    while((c=getchar())!=EOF)
    {
        if(((!CHANGE)&&flag2) || flag1)
        {
            putchar(c);
            flag1=1;
        }
        else
        {
            if(c>=97&&c<=122)putchar(c-32);
            else if(c>=65&&c<=90)putchar(c+32);
            else putchar(c);
        }
        flag2=0;
    }
}
```

```

        if(c=='\n') {flag1=0;flag2=1;}
    }
}

```

5 数组实验

5.1 实验目的

- (1) 掌握数组的说明、初始化和使用。
- (2) 掌握一维数组作为函数参数时实参和形参的用法。
- (3) 掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。
- (4) 掌握基于分治策略的二分查找算法和选择法排序算法的思想，以及相关算法的实现。

5.2 实验内容及要求

5.2.1 源程序改错

下面是用来将数组 **a** 中元素按升序排序后输出的源程序。分析源程序中存在的问题，并对源程序进行修改，使之能够正确完成任务。

源程序

```

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 }

```

解答：

（1）错误修改：

51.c:7:10: 警告：传递'sort'的第 1 个参数时将整数赋给指针，未作类型转

换 [-Wint-conversion] sort(a[0],10);

line7: sort(a,10);

sort()内疑似逻辑错误，直接重写。

（2）错误修改后运行结果：

```

recolic@RECOLICPC ~/c/h/s/17c> make
gcc --version
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权声明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
uname -a
Linux RECOLICPC 4.10.10-1-ARCH #1 SMP PREEMPT Wed Apr 12 18:50:28 CEST 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/s/17c> make 1
cat 51.c
#include<stdio.h>
int main(void)
{
    int a[10] = {27, 13, 5, 32, 23, 3, 17, 43, 55, 39};
    void sort(int [],int);
    int i;
    sort(a,10);
    for(i = 0; i < 10; i++)
        printf("%6d",a[i]);
    printf("\n");
    return 0;
}

#include <stdlib.h>
int _is_less(const void *elea, const void *eleb) {return *(const int *)elea - *(const int *)eleb;}

void sort(int b[], int n)
{
    qsort((void *)b, n, sizeof(int), _is_less);
}

/*
    int i, j, t;
    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - i - 1; j++)
            if(b[j] > b[j+1])
                t = b[j], b[j] = b[j+1], b[j+1] = t;
*/
gcc 51.o -o 51
./51
    3    5   13   17   23   1 27 32 3 17 43 55

```

5 数组实验

5.1 实验目的

- (1) 掌握数组的说明、初始化和使用。
- (2) 掌握一维数组作为函数参数时实参和形参的用法。
- (3) 掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。

5.2 实验内容及要求

5.2.1 源程序改错

下面是用来将数组 a 中元素按升序排序后输出的源程序。分析源程序中存在的问题，并对源程序进行修改，使之能够正确完成任务。

源程序

5.2.2 源程序完善、修改、替换

(1) 下面的源程序用于求解瑟夫问题：M 个人围成一圈，从第一个人开始依次从 1 至 N 循环报数，每当报数为 N 时报数人出圈，直到圈中只剩下一个人为止。请在源程序中的下划线处填写合适的代码来完善该程序。

源程序：

```

#include<stdio.h>
#define M 10
#define N 3
int main(void)
{
    int a[M], b[M]; /* 数组 a 存放圈中人的编号，数组 b 存放出圈人的编号 */
    int i, j, k;
    for(i = 0; i < M; i++) /* 对圈中人按顺序编号 1—M */
        a[i] = i + 1;

```



```

        for(i = M, j = 0; i > 1; i--){
/* i 表示圈中人数，初始为 M 个，剩 1 个人时结束循环；j 表示当前报数人的位置 */
        for(k = 1; k <= N; k++)          /* 1 至 N 报数 */
            if(++j > i - 1) j = 0; /* 最后一个人报数后第一个人接着报，形成一个圈 */
            b[M-i] = j? j : 0 ; /* 将报数为 N 的人的编号存入数组 b */
            if(j)
                for(k = --j; k < i; k++) /* 压缩数组 a，使报数为 N 的人出圈 */
                    a[k]=a[k+1];
        }
        for(i = 0; i < M - 1; i++) /* 按次序输出出圈人的编号 */
            printf("%6d", b[i]);
        printf("%6d\n", a[0]); /* 输出圈中最后一个人的编号 */
    return 0;
}

```

(2) 上面的程序中使用数组元素的值表示圈中人的编号，故每当有人出圈时都要压缩数组，这种算法不够精炼。如果采用做标记的办法，即每当有人出圈时对相应数组元素做标记，从而可省掉压缩数组的时间，这样处理效率会更高一些。因此，请采用做标记的办法修改（1）中的程序，并使修改后的程序与（1）中的程序具有相同的功能。

```

#define _nr_M 10
#define _nr_N 3

#include <rlib/io.hpp>

#include <iostream>
#include <deque>
#include <algorithm>
#define M _nr_M
#define N _nr_N
using namespace std;
using namespace recolic_hpp;

int main()
{
    deque<int> quBuffer;
    quBuffer.resize(M);
    for_each(quBuffer.begin(), quBuffer.end(), [](int &ri){static int cter = 0;ri=++cter;});
    size_t currPos = (size_t)(-1);
    while(quBuffer.size() > 1)
    {
        size_t posToErase = (currPos + N) % quBuffer.size();
        currPos = posToErase - 1;
        println("To erase:", quBuffer[posToErase]);
        quBuffer.erase(quBuffer.begin() + posToErase);
    }
    println("Result=", quBuffer[0]);
}

```

```

    return 0;
}

```

给出的原程序可读性差，已完全重写。

```

recolic@RECOLICPC ~/c/h/s/17c> make
gcc --version
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权声明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
uname -a
Linux RECOLICPC 4.10.10-1-ARCH #1 SMP PREEMPT Wed Apr 12 18:50:28 CEST 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/s/17c> make 22.o.hpp
g++ 522.o -o 522
./522
To erase: 3
To erase: 6
To erase: 9
To erase: 2
To erase: 7
To erase: 1
To erase: 8
To erase: 5
To erase: 10
Result= 4

```

5.2.3 跟踪调试源程序

在下面所给的源程序中，函数 `strncat(s,t,n)` 本来应该将字符数组 `t` 的前 `n` 个字符连接到字符数组 `s` 中字符串的尾部。但函数 `strncat` 在定义时代码有误，不能实现上述功能。请按下面的要求进行操作，并回答问题和排除错误。

(1) 单步执行源程序。进入函数 `strncat` 后观察表达式 `s`、`t` 和 `i`。当光条落在 `for` 语句所在行时，`i` 为何值？当光条落在 `strncat` 函数块结束标记（右花括号 `}`）所在行时，`s`、`t` 分别为何值？

(2) 分析函数出错的原因，排除错误，使函数正确实现功能，最后写出程序的输出结果。

源程序：

```

#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';
}

```

```

(1) (gdb) print i    $1 = 23
(gdb) print s
$2 = 0x7fffffffe110 "The adopted symbol is "
(gdb) print t
$3 = 0x7fffffffe0f0 "abcdefghijklmnopqrstuvwxyz"

```

(2) The adopted symbol is abcd

5.2.4 程序设计

编写并上机调试运行能实现以下功能的程序。

(1) 编写一个程序,从键盘读取数据, 对一个 3(4 矩阵进行赋值, 求其转置

矩阵, 然后输出原矩阵和转置矩阵。

```

#include <stdio.h>
#define row 3
#define column 4
int a[column][row];
void Transposedmatrix(int matrix[][column], int m, int n);
int main(void)
{
    int matrix[row][column], i, j;
    for (i = 0; i < row; i++)
        for (j = 0; j < column; j++)
            scanf("%d", &matrix[i][j]);
    for (i = 0; i < row; i++) {
        for (j = 0; j < column; j++)
            printf("%5d", matrix[i][j]);
        putchar('\n');
    }
    putchar('\n');
    Transposedmatrix(matrix, row, column);
    for (i = 0; i < column; i++) {
        for (j = 0; j < row; j++)
            printf("%5d", a[i][j]);
        putchar('\n');
    }
}

```

```

        return 0;
    }
    void Transposedmatrix(int matrix[][column ], int m, int n)
    {
        int i, j;
        for (i = 0; i < n; i++)
            for (j = 0; j < m; j++)
                a[i][j] = matrix[j][i];
    }

```

(2) 编写一个程序，其功能要求是：输入一个整数，将它在内存中二进制表示的每一位转换成为对应的数字字符，存放到一个字符数组中，然后输出该整数的二进制表示。

```

#include<stdio.h>
int main(void)
{
    int N;
    int n;
    char two[33];
    sizeof(int);
    scanf("%d", &N);
    while(N--)
    {
        int i, j;
        scanf("%d", &n);
        if(n>0)
        {
            two[0] = '0';
            for(i=0; n!=0; i++)
            {
                two[31-i] = n%2 + '0';
                n /= 2;
            }
            for(j = 31-i; j>=0; j--)
                two[j] = '0';
            two[32] = 0;
            printf("%s\n", two);
        }
        else
        {
            for(i=0; i<=31; i++)
            {
                two[31-i] = (((n & (0x1 << i)) >> i) & 0x1) + '0';
            }
            two[32] = 0;
            printf("%s\n", two);
        }
    }
    return 0;
}

```

(3) 编写一个程序，其功能要求是：输入 n 个学生的姓名和 C 语言课程的成绩，将成绩按从高到低的次序排序，姓名同时作相应调整，输出排序后学生的姓名和 C 语言课程的成绩。然后，输入一个 C 语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，输出该成绩同学的姓名和 C 语言课程的成绩；否则输出提示 “not found!”。

```
#include<stdio.h>
#include<string.h>
#define N 100
void sort(char name[][N],int score[],int n);
int binary_search(int score[],int n,int x);
int main(void)
{
    //freopen("data.in","r",stdin);
    //freopen("data.out","w",stdout);
    int i,n,x,m,j;
    int score[N];
    char name[N][N];

    scanf("%d",&n);
    //printf("right\n");
    //for(i=0;i<n;i++)
        //name[i]=(char *)malloc(20);
    //printf("right\n");

    for(i=0;i<n;i++)
        scanf("%s%d",name[i],score+i);
    sort(name,score,n);
    //printf("right\n");
    for(i=0;i<n;i++)
        printf("%-21s%d\n",name[i],score[i]);
    printf("\n");

    scanf("%d",&m);
    for(j=0;j<m;j++)
    {
        scanf("%d",&x);
        if((i=binary_search(score,n,x))>=0)
            printf("%-21s%d\n",name[i],score[i]);
        else
            printf("Not found!\n");
    }

    return 0;
}
void sort(char name[][N],int score[],int n)
{
    int i,j,t;
    char tem[1][N];
    for(i=0;i<n-1;i++)
        for(j=0;j<n-i-1;j++)
```

```

        if(score[j]<score[j+1])
        {
            t=score[j],score[j]=score[j+1],score[j+1]=t;
        }
        strcpy(tem[0],name[j]),strcpy(name[j],name[j+1]),strcpy(name[j+1],tem[0]);
    }
}
int binary_search(int score[],int n,int x)
{
    int middle,front=0,back=n-1;
    while(front<=back)
    {
        middle=(front+back)/2;
        if(x<score[middle])
            front=middle+1;
        else if(x>score[middle])
            back=middle-1;
        else
            return middle;
    }
    return -1;
}

```

5.Optional question:

(1)

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
char *strnins(char *dst, const char *src, size_t n)
{
    //Dangerous function.
    size_t bufSize = strlen(dst) + strlen(src) + 1;
    char *tmpBuffer = (char *)malloc(sizeof(char) * bufSize);
    tmpBuffer[bufSize] = '\0';

    char eatenChar = dst[n];
    dst[n] = '\0';
    sprintf(tmpBuffer, "%s%s%c%s", dst, src, eatenChar, (char *) (dst + n +
1));

    strcpy(dst, tmpBuffer);
    free(tmpBuffer);
    return dst;
}

int main()
{
    char dst[1024] = "aaaaaaaaaaaaaaaaaaaaa";
    printf("%s", strnins(dst, "123", 2));
    return 0;
}

```

```

recolic@RECOLICPC ~/c/h/s/17c> make
gcc --version
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权声明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
uname -a
Linux RECOLICPC 4.10.10-1-ARCH #1 SMP PREEMPT Wed Apr 12 18:50:28 CEST 2017; x86_64; GNU/Linux
recolic@RECOLICPC ~/c/h/s/17c> make 4
gcc 54.o -o 54
./54
aa123aaaaaaaaaaaaaaaaaa

```

(2) multi-queen puzzle:(Backtrack solution)

```

#include <iostream>
using namespace std;

```

```

const int N = 8;
int position[N];

```

```

// Check if a position is safe
bool isSafe(int queen_number, int row_position)
{
    // Check each queen before this one
    for (int i = 0; i < queen_number; i++)
    {
        // Get another queen's row_position
        int other_row_pos = position[i];

        // Now check if they're in the same row or diagonals
        if (other_row_pos == row_position || // Same row
            other_row_pos == row_position - (queen_number - i) || // Same
diagonal
            other_row_pos == row_position + (queen_number - i)) // Same
diagonal
            return false;
    }
    return true;
}

```

// Recursively generate a tuple like [0 0 0 0], then [0 0 0 1] then etc...

```

void solve(int k)
{
    if (k == N) // We placed N-1 queens (0 included), problem solved!
    {
        // Solution found!
        cout << "Solution: ";
        for (int i = 0; i < N; i++)
            cout << position[i] << " ";
        cout << endl;
    }
    else
    {
        for (int i = 0; i < N; i++) // Generate ALL combinations
        {
            // Before putting a queen (the k-th queen) into a row, test it for
safeness

```

```

        if (isSafe(k, i))
        {
            position[k] = i;
            // Place another queen
            solve(k + 1);
        }
    }
}

int main()
{
    solve(0);

    return 0;
}

```

实验 6 指针实验

一、实验目的

1. 熟练掌握指针的说明、赋值、使用。
2. 掌握用指针引用数组的元素，熟悉指向数组的指针的使用。
3. 熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。
4. 掌握指针函数与函数指针的用法。
5. 掌握带有参数的 main 函数的用法。

二、实验题目及要求

1. 源程序改错题

下面程序是否存在错误？如果存在，原因是什么？如果存在错误，要求在计算机上对这个例子程序进行调试修改，使之能够正确执行。


```
#include "stdio.h"
#include <stdlib.h>
void main(void)
{
    float *p = malloc(sizeof(float));
    scanf("%f",p);
    printf("%f\n",*p);
    free(p);
}
解答：
```

(1) 错误修改：

61.c:5: 未分配空间就使用指针存储数据。

未包含 stdlib.h

(2) 错误修改后运行结果



```
recolic@RECOLICPC ~/c/h/s/17c> make
gcc --version
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权说明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保要求
uname -a
Linux RECOLICPC 4.10.13-1-ARCH #1 SMP PREEMPT Thu Apr 27 12:15:09 CEST 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/s/17c> make
gcc 61.o -o 61
./61
12.34
12.340000
```

2. 源程序完善、修改、替换题

(1) 下面的程序通过函数指针和菜单选择来调用字符串拷贝函数或字符串

连接函数，请在下划线处填写合适的表达式、语句、或代码片段来完善该程序。

```
#include "stdio.h"
#include "string.h"
void main(void)
{
    _____ typedef (strcpy) *p;
    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\tinput 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;
    scanf("%s", a); //Warning: Dangerous operation!
printf("input the second string please!\n");
i=0;
    scanf("%s", b); //Warning: Dangerous operation!
result= p (a,b);
printf("the result is %s\n",result);
down:
    ;
}

```

(2) 为了使程序不受 scanf、getchar、gets 等函数输入后回车符的影响，请修改第(1)题程序，按要求输出下面结果：((输入)表示该数据是键盘输入数据)

```

1 copy string.
2 connect string.
3 exit.
input a number (1-3) please!

```

2 (输入)

```

input the first string please!
the more you learn, (输入)
input the second string please!
the more you get. (输入)
the result is the more you learn,the more you get.

```

解答:

```

#include "stdio.h"
#include "string.h"
#include <stdlib.h>
void fuck(char *sa)
{ //Dangerous Function!
    __typeof__(sa) s = sa;
    while(*s)
    {
        ++s; //Warning: logic error.
        if(*s=='\n')
            *s = 0;
    }
}
int main(void)
{
    __typeof__(strcpy) *p;
    char *a = malloc(1024),*b=malloc(1024),c[1600],*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\tinput 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;
    FILE *in = fdopen(0,"r");
    size_t nullLen = 1024;
    getline(&a,&nullLen,in);
    fuck(a);
    printf("input the second string please!\n");
    i=0;
    getline(&b,&nullLen,in);
    fuck(b);
    fclose(in);
    sprintf(c,"%s%s",a,b);
    free(a);free(b);
    printf("the result is %s\n",result);
down:
    ;
    return 0;
}

```

```

recolic@RECOLICPC ~/c/h/s/17c> make
gcc --version
gcc (GCC) 6.3.1 20170306
Copyright © 2016 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权声明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
uname -a
Linux RECOLICPC 4.10.13-1-ARCH #1 SMP PREEMPT Thu Apr 27 12:15:09 CEST 2017 x86_64 GNU/Linux
recolic@RECOLICPC ~/c/h/s/17c> make 22
g++ 622.o -o 622
./622
1 copy string.
2 connect string.
3 exit.
input a number (1-3) please!
2
input the first string please!
1 2 3 4 5 6,
input the second string please!
7 8 9 1 2 3 ,
the result is 1 2 3 4 5 6,7 8 9 1 2 3

```

3. 跟踪调试题

```
#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) 单步执行。进入 strcpy 时 watch 窗口中 s 为何值？返回 main 时, watch 窗口中 s 为何值？

(2) 排除错误，使程序输出结果为：

there is a boat on the lake.

(3) 选做：由于 watch 窗口中只显示 s 所指串的值，不显示 s 中存储的地址值，怎样才能观察到 s 值的变化呢？

```

(1)
(gdb) break 9
Breakpoint 1 at 0x4005a9: file 63.c, line 9.
(gdb) break 12
Breakpoint 2 at 0x4005ce: file 63.c, line 12.
(gdb) run
Starting program: /home/recolic/cpp/homework/src/17c/63.o.dbg

Breakpoint 1, strcpy (s=0x7fffffffe030 "\340\005@", t=0x7fffffffdff0 "there is a
boat on the lake.") at 63.c:10
10     while(*s++=*t++)
(gdb) print s
$1 = 0x7fffffffe030 "\340\005@"
(gdb) continue
Continuing.

Breakpoint 2, strcpy (s=0x7fffffffe04d "", t=0x7fffffffe00d "") at 63.c:12
12     return (s);
(gdb) print s
$2 = 0x7fffffffe04d ""
(2)
#include "stdio.h"
char *strcpy(char *,char *);
int main(void)
{
    char a[1024],b[1024]="there is a boat on the lake.";
    printf("%s\n",strcpy(a,b));
    return 0;
}

```

```

char *strcpy(char *s,char *t)
{ //Extremely Dangerous!
    char *backup = s;
    while(1)
    {
        if(*t == '\0')
            return backup;
        *s = *t;
        ++s, ++t;
    }
    return backup;
}

```

(3)

printf 调试法。

4. 编程设计题

(1) 一个长整型变量占 4 个字节，其中每个字节又分成高 4 位和低 4 位。

试从该长整型变量的高字节开始，依次取出每个字节的高 4 位和低 4 位并以数字字符的形式进行显示。

```

#include <stdio.h>
int main() {
    int c,sp=0;
    scanf("%d", &c);
    long n;
    for (int i = 0; i < c;i++){
        scanf("%ld", &n);
        long i;
        char *p = (char *)&n;
        char low, up;
        for (i = 3; i >= 0; i--) {
            up = (p[i]) & 0x0F;
            low = (p[i] >> 4) & 0x0F;
            if (low>9) {
                putchar(low - 10 + 'A');
                if (sp < 4) {
                    putchar(' ');
                    sp++;
                }
            }
            else {
                sp = 0;
            }
        }
        else {
            if (sp < 4) { printf("%d ", low); sp++; }
            else {
                printf("%d", low);
            }
        }
    }
}

```

```

        sp = 0;
    }
}
if (up>9) {
    putchar(up - 10 + 'A');
    if (sp < 4) { putchar(' ');
    }
    else sp = 0;
}
else {
    if (sp < 4) { printf("%d ", up); }
    else { sp = 0; printf("%d", up); }
}
}
putchar('\n');
}
return 0;
}

```

(2) 利用大小为 n 的指针数组指向用 `gets` 函数输入的 n 行，每行不超过 80 个字符。编写一个函数，它将每一行中连续的多个空格字符压缩为一个空格字符。在调用函数中输出压缩空格后的各行，空行不予输出。

```

#include<stdio.h>
#include<ctype.h>
void reduce(char *s);
int main()
{
    int k,j,n,i;
    char a[10][81];
    char *p=&a[0][0];
    while(scanf("%d", &n),n!=0)
    {
        for(k=0;k<=n-1;k++)
        {
            fgets(p+k,100,stdin);
        }
        for(i=0;i<=n-1;i++)
        {
            reduce((p+i));
            printf("%s\n",a[i]);
        }
    }
    return 0;
}
void reduce(char*s)
{
    int i,j;
    for(i=0,j=0;s[i]!='\0';i++,j++)
    {
        if(isspace(s[i]))
        {
            for(;isspace(s[i]);i++) ;
            s[j++]=' ';
            s[j]=s[i];
        }
    }
}

```

```

    }
    else s[j]=s[i];
  }
  s[j]=s[i];
}

```

(3) 设某个班有 N 个学生，每个学生修了 M 门课程（用#define 定义 N、M）。输入 M 门课程名称，然后依次输入 N 个学生中每个学生所修的 M 门课程的成绩并且都存放到相应的数组中。编写下列函数：

- 计算每个学生各门课程平均成绩；
- 计算全班每门课程的平均成绩；
- 分别统计低于全班各门课程平均成绩的人数；
- 分别统计全班各门课程不及格的人数和 90 分以上（含 90 分）的人数。

在调用函数中输出上面各函数的计算结果。（要求都用指针操作，不得使用下标操作。）

```

#include<stdio.h>
#define N 5
#define M 5
void mgrdpjcj(float (*p)[M],char (*q)[10]);
void mmkdpjcj(float (*p)[M],char (*q)[10]);
void tjdyjfrs(float (*p)[M],char (*q)[10]);
float kcpjcj[M];
int main(void)
{
    char kmm[M][10],xsm[N][10];
    float fs[N][M];
    int i,j;
    for(i=0;i<M;i++)
    {
        scanf("%s",kmm[i]);
    }
    for(i=0;i<N;i++)
    {
        scanf("%s",xsm[i]);
        for(j=0;j<M;j++)
        {
            scanf("%f",&fs[i][j]);
        }
    }
    mgrdpjcj(fs,xsm);
    mmkdpjcj(fs,kmm);
    tjdyjfrs(fs,kmm);
    return 0;
}

```

```

}
void mgrdpjcg(float (*p)[M],char (*q)[10])
{
    int i,j;
    float sum;
    for(i=0;i<N;i++)
    {
        for(j=0,sum=0;j<M;j++)
        {
            sum+=*(*p+i)+j);
        }
        printf("Average score of %s is %.2f\n",*(q+i),sum/M);
    }
}
void mmkdpjcg(float (*p)[M],char (*q)[10])
{
    int i,j;
    float sum;
    for(i=0;i<M;i++)
    {
        for(j=0,sum=0;j<N;j++)
        {
            sum+=*(*p+j)+i);
        }
        printf("Average score of %s is %.2f\n",*(q+i),kcpjcg[i]=sum/N);
    }
}
void tjdpjfrs(float (*p)[M],char (*q)[10])
{
    int i,j,k;
    for(i=0;i<M;i++)
    {
        for(j=0,k=0;j<N;j++)
        {
            if(*(*p+j)+i)<kcpjcg[i])
                k++;
        }
        printf("Number of students lower than avg of %s is %d\n",*(q+i),k);
    }
    for(i=0;i<M;i++)
    {
        for(j=0,k=0;j<N;j++)
        {
            if(*(*p+j)+i)<60)
                k++;
        }
        printf("Number of students %s fail is %d\n",*(q+i),k);
    }
    for(i=0;i<M;i++)
    {
        for(j=0,k=0;j<N;j++)
        {
            if(*(*p+j)+i)>=90)
                k++;
        }
        printf("Number of students %s perfect is %d\n",*(q+i),k);
    }
}
}

```


5. 选做题

(1) 设有 N 位整数和 M 位小数 (N=20, M=10) 的数据 a,b。编程计算 a+b 并输出结果。

如: 12345678912345678912.1234567891 +

98765432109876543210.0123456789

```
#include <unistd.h>
#include <stdlib.h>
int main(){
system("echo \"#!/bin/env python3\" > /tmp/66.py");
system("echo \"a,b=input('Give a+b >').split('+')\" >> /tmp/66.py");
system("echo \"print(float(a)+float(b))\" >> /tmp/66.py");
system("chmod +x /tmp/66.py");
return execl("/tmp/66.py","");}
```

(2) 编写使用复杂声明 `char *(*p[2])(const char *,const char *)`;的程序。

提示: p 中元素可为 `strcmp`、`strstr` 等函数名。

实验 7 结构与联合实验

一、实验目的

1. 通过实验,熟悉和掌握结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。
2. 通过实验,掌握动态储存分配函数的用法,掌握自引用结构,单向链表的创建、遍历、结点的增删、查找等操作。
3. 了解字段结构和联合的用法。

二、实验题目及要求

1. 表达式求值的程序验证题

设有说明：

```
char u[]="UVWXYZ";
```

```
char v[]="xyz";
```

```
struct T{
```

```
    int x;
```

```
    char c;
```

```
    char *t;
```

```
}a[]={{{11, 'A', u},{100, 'B', v}},*p=a;
```

请先自己计算下面表达式的值，然后通过编程计算来加以验证。(各表达式相互无关)

序号	表达式	计算值	验证值
1	(++p)->x	100	100
2	p++,p->c	B	B
3	*p++->t,*p->t	x	x
4	*(++p)->t	x	x
5	*++p->t	V	V
6	++*p->t	V	V

2. 源程序修改替换题

给定一批整数，以 0 作为结束标志且不作为结点，将其建成一个先进先出的链表，先进先出链表的指头指针始终指向最先创建的结点（链头），先建结点指向后建结点，后建结点始终是尾结点。

源程序中存在什么样的错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。

源程序如下：

```

#include "stdio.h"
#include "stdlib.h"
struct s_list{
int data; /* 数据域 */
struct s_list *next; /* 指针域 */
};
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 为结束标记 */
    create_list(head,s); /* 创建新链表 */
    p=head; /*遍历指针 p 指向链头 */
    while(p){
        printf("%d\t",p->data); /* 输出数据域的值 */
        p=p->next; /*遍历指针 p 指向下一结点 */
    }
    printf("\n");
}
void create_list(struct s_list *headp,int *p)
{
    struct s_list * loc_head=NULL,*tail;
    if(p[0]==0) /* 相当于*p==0 */
        ;
    else { /* loc_head 指向动态分配的第一个结点 */
        loc_head=(struct s_list *)malloc(sizeof(struct s_list));
        loc_head->data=*p++; /* 对数据域赋值 */
        tail=loc_head; /* tail 指向第一个结点 */
        while(*p){ /* tail 所指结点的指针域指向动态创建的结点 */
            tail->next=(struct s_list *)malloc(sizeof(struct s_list));
            tail=tail->next; /* tail 指向新创建的结点 */
            tail->data=*p++; /* 向新创建的结点的数据域赋值 */
        }
        tail->next=NULL; /* 对指针域赋 NULL 值 */
    }
    headp=loc_head; /* 使头指针 headp 指向新创建的链表 */
}

```

```

#include<stdio.h>
#include<stdlib.h>
struct s_list{
int data;
    struct s_list *next;
};
void creat_list(struct s_list **headp, int *p);
int main(void) {
    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;
    if(p[0] == 0)
        ;
    else{
        loc_head = (struct s_list *)malloc(sizeof(struct s_list));
        loc_head -> data = *p++;
        tail = loc_head;
        while(*p) {
            tail -> next = (struct s_list *)malloc(sizeof(struct s_list));
            tail = tail -> next;
            tail -> data = *p++; }
        tail -> next = NULL; }
    *headp = loc_head;
}

```

(2) 修改替换 create_list 函数，将其建成一个后进先出的链表，后进先出链表的头指针始终指向最后创建的结点（链头），后建结点指向先建结点，先建结点始终是尾结点。

```

#include <stdio.h>
#include <stdlib.h>
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;
}

```

3. 编程设计题

(1) 设计一个字段结构 `struct bits`，它将一个 8 位无符号字节从最低位向高位声明为 8 个字段，各字段依次为 `bit0`, `bit1`, ..., `bit7`，且 `bit0` 的优先级最高。同时设计 8 个函数，第 `i` 个函数以 `biti(i=0,1,2,...,7)` 为参数，并且在函数体内输出 `biti` 的值。将 8 个函数的名字存入一个函数指针数组 `p_fun`。如果 `bit0` 为 1，调用 `p_fun[0]` 指向的函数。如果 `struct bits` 中有多位为 1，则根据优先级从高到低依次调用函数指针数组 `p_fun` 中相应元素指向的函数。8 个函数中的第 0 个函数可以设计为：

```

        void f0(struct bits b)
        {
            Printf("the function %d is called!\n",b);
        }

#include<stdio.h>
#include<stdlib.h>
struct ISR_BITS
{
    unsigned int bit0:1;
    unsigned int bit1:1;
    unsigned int bit2:1;
    unsigned int bit3:1;
    unsigned int bit4:1;
    unsigned int bit5:1;
    unsigned int bit6:1;
    unsigned int bit7:1;
    unsigned int rsv:8;
};
void isr0()
{
    printf("The Interrupt Service Routine isr0 is called!\n");
}

```

```

void isr1()
{
    printf("The Interrupt Service Routine isr1 is called!\n");
}
void isr2()
{
    printf("The Interrupt Service Routine isr2 is called!\n");
}
void isr3()
{
    printf("The Interrupt Service Routine isr3 is called!\n");
}
void isr4()
{
    printf("The Interrupt Service Routine isr4 is called!\n");
}
void isr5()
{
    printf("The Interrupt Service Routine isr5 is called!\n");
}
void isr6()
{
    printf("The Interrupt Service Routine isr6 is called!\n");
}
void isr7()
{
    printf("The Interrupt Service Routine isr7 is called!\n");
}
int main()
{
    union ISR_REG
    {
        unsigned short all;
        struct ISR_BITS bit;
    } isr_reg;
    int i,n;
    unsigned int a[100];
    scanf("%d",&n);
    for(i=0; i<n; i++)
        scanf("%u",&a[i]);
    for(i=0; i<n; i++)
    {
        isr_reg.all=a[i];
        printf("%u:\n",a[i]);
        void (*p_isr[8])();
        p_isr[0]=isr0;
        p_isr[1]=isr1;
        p_isr[2]=isr2;
        p_isr[3]=isr3;
        p_isr[4]=isr4;
        p_isr[5]=isr5;
        p_isr[6]=isr6;
        p_isr[7]=isr7;
        if(isr_reg.bit.bit0) p_isr[0]();
        if(isr_reg.bit.bit1) p_isr[1]();
        if(isr_reg.bit.bit2) p_isr[2]();
        if(isr_reg.bit.bit3) p_isr[3]();
        if(isr_reg.bit.bit4) p_isr[4]();
    }
}

```

```

        if(isr_reg.bit.bit5) p_isr[5]();
        if(isr_reg.bit.bit6) p_isr[6]();
        if(isr_reg.bit.bit7) p_isr[7]();
        printf("\n");
    }
    return 0;
}

```

(2) 用单向链表建立一张班级成绩单，包括每个学生的学号、姓名、英语、高等数学、普通物理、C 语言程序设计四门课程的成绩。用函数编程实现下列功能：

- (1) 输入每个学生的各项信息。
- (2) 输出每个学生的各项信息。
- (3) 修改指定学生的指定数据项的内容。
- (4) 统计每个同学的平均成绩（保留 2 位小数）。
- (5) 输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct s_list
{
    char num[15];
    char name[20];
    float English,Math,Physics,C;
    float total,avg;
    struct s_list *next;
}*p,*q;
struct s_list *input(int n)
{
    int i;
    struct s_list *head = NULL;
    head=(struct s_list *)malloc(sizeof(struct s_list));
    for(i=0; i<n; i++)
    {
        p=(struct s_list *)malloc(sizeof(struct s_list));
        if(i==0)
            head=p;
        else
            q->next=p;
        scanf("%s%s",p->num,p->name);
        scanf("%f%f%f%f",&p->English,&p->Math,&p->Physics,&p->C);
    }
}

```

```

        q=p;
        p->next=NULL;
    }
    return head;
}
void display(struct s_list *head,int n)
{
    p=(struct s_list *)malloc(sizeof(struct s_list));
    for (p=head; n>0; n--,p=p->next)
    {
        printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",p->num,p->name,p
->English,p->Math,p->Physics,p->C);
    }
    printf("\n");
}
void change(struct s_list *head,int n)
{
    char a[10][15],b[10][20];
    int d,i,j,m;
    float c[10];
    scanf("%d",&m);
    for(i=0; i<m; i++)
        scanf("%s%s%f",a[i],b[i],&c[i]);
    for(i=0; i<m; i++)
    {
        for(q=head,j=n; j>0; j--,q=q->next)
        {
            if(!(strcmp(q->num,a[i])))
            {
                if(!(strcmp("English",b[i]))) q->English=c[i];
                else if(!(strcmp("Math",b[i]))) q->Math=c[i];
                else if(!(strcmp("Physics",b[i]))) q->Physics=c[i];
                else if(!(strcmp("C",b[i]))) q->C=c[i];
            }
        }
    }
}
void f1(struct s_list *head,int n)
{
    float a,b;
    int i,j;
    q=(struct s_list*)malloc(sizeof(struct s_list));
    q=head;
    for(i=0; i<n; i++)
    {
        a=(q->English)+(q->Math)+(q->Physics)+(q->C);
        q->total=a;
        b=a/4;
        q->avg=b;
        q=q->next;
    }
    printf("SumAndAvg:\n");
    printf("%-15s%-20s%-10s%-10s\n","ID","Name","SUM","AVG");
    for (q=head; n>0; n--,q=q->next)
    {
        printf("%-15s%-20s%-10.2f%-10.2f\n",q->num,q->name,q->total,q->avg);
    }
    printf("\n");
}

```



```

}
void sort(struct s_list *head,int n)
{
    struct s_list *p1=head,*p2;
    char a[10][15],b[10][20];
    int i,j;
    float t;
    for(i=0,p1=head; i<n-1; i++,p1=p1->next)
    {
        for(j=i+1,p2=p1->next; j<n; j++,p2=p2->next)
        {
            if(p1->avg>p2->avg)
            {
                strcpy(a[0],p1->num);
                strcpy(p1->num,p2->num);
                strcpy(p2->num,a[0]);
                strcpy(b[0],p1->name);
                strcpy(p1->name,p2->name);
                strcpy(p2->name,b[0]);
                t=p1->avg;
                p1->avg=p2->avg;
                p2->avg=t;
            }
        }
    }
    printf("Sort:\n");
    printf("%-15s%-20s%-10s\n","ID","Name","AVG");
    for (p=head; n>0; n--,p=p->next)
    {
        printf("%-15s%-20s%-10.2f\n",p->num,p->name,p->avg);
    }
    printf("\n");
}
int main()
{
    int n;
    struct s_list *head;
    scanf("%d",&n);
    head=input(n);
    printf("%-15s%-20s%-10s%-10s%-10s%-10s\n","ID","Name","English","Math","Physics","C");
    display(head,n);
    change(head,n);
    printf("Alter:\n");
    printf("%-15s%-20s%-10s%-10s%-10s%-10s\n","ID","Name","English","Math","Physics","C");
    display(head,n);
    f1(head,n);
    sort(head,n);
    return 0;
}

```

4. 选做题

(1) 对编程设计题第(2)题的程序, 增加按照平均成绩进行升序排序的函数, 写出用交换结点数据域的方法升序排序的函数, 排序可用选择法或冒泡法。

`std::algorithm` 不是吃素的。

(2) 对选做题第(1)题, 进一步写出用交换结点指针域的方法升序排序的函数。

`list_node` 是支持 `std::move` 的。

(3) 采用双向链表重做编程设计题第(2)题。

仅 C++。

8 文件实验

8.1 实验目的

1. 熟悉文本文件和二进制文件在磁盘中的存储方式;
2. 熟练掌握流式文件的读写方法。

8.2 实验题目及要求

8.2.1. 文件类型的程序验证题

设有程序:

```
#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;
}

```

请思考程序的输出结果，然后通过上机运行来加以验证。

写入一些内容到文件'.d:\abc1.bin' '.d:\abc2.txt'(小心转义)。

将两处 `sizeof(short)`均改为 `sizeof(char)`结果有什么不同，为什么？

写入字节数不同。原因见 `man fwrite`。

(3) 将 `fprintf(fp2,"%hx %hx",a,b)` 改为 `fprintf(fp2,"%d %d",a,b)`结果有什么不同。

`fp2` 内容发生变化。

8.2.2. 源程序修改替换题

将指定的文本文件内容在屏幕上显示出来，命令行的格式为：

`type filename`

源程序中存在什么样的逻辑错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。

```

#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 指向 filename */
    printf("Can't open %s file!\n",argv[1]);
    exit(-1);
}

while(ch=fgetc(fp)!=EOF)                    /* 从 filename 中读字符 */
    putchar(ch);                            /* 向显示器中写字符 */
fclose(fp);                                /* 关闭 filename */
return 0;
}

```

(2) 用输入输出重定向 `freopen` 改写上述源程序中的 `main` 函数。

```

#include <unistd.h>
int main(int ac, char **argv) {return execl("/usr/bin/cat",argv);}

```

3. 编程设计题

(1) 从键盘输入一行英文句子，将每个单词的首字母换成大写字母，然后

输出到一个磁盘文件“test”中保存。

```

#include <unistd.h>
#include <stdlib.h>
int main(){
    system("echo '#!/bin/bash' > /tmp/73.sh");
    system("echo 'xargs -n 1 | sed 's/.*/&u&' | xargs' >> /tmp/73.sh");
    system("chmod +x /tmp/73.sh");
    return execl("/tmp/73.sh","");
}

```

参考文献

C11 standard (ISO/IEC 9899:2011)

C99 standard (ISO/IEC 9899:1999)

C89/C90 standard (ISO/IEC 9899:1990)

Linux man pages online <https://linux.die.net/man/>

<http://man7.org/linux/man-pages/>

C/C++ Standard Library Documentation <http://www.cplusplus.com/reference/>

<http://en.cppreference.com/w/>

Python 3.6 Documentation <https://docs.python.org/3/>