通知

- 答疑：　2019-12-25　16:00-18:00　计算机楼418
  　　2019-12-31　16:00-18:00　基础实验楼乙124
- 闭卷笔试：　2020-1-2　16:30-18:30　？（自行关注系统）
- 样卷已上传至课程网站

关于输入/输出（只需掌握C/C++任意一种）
关于库函数（如果需要，会给出函数原型和头文件）

- **成绩：**
  - 部分平时作业、第三次测验及扣分原因、期末笔试成绩将陆续导入**课程网站**（四舍五入）；
  - 需要详细成绩的可以在课程网站成绩上传后（考试周第二周）找助教老师要 Excel 成绩条；
  - 部分平时作业、前两次测验成绩及扣分原因已导入**课程网站**（四舍五入）；
  - **请在2020年1月10号 17:00 之前仔细核对，并反馈。**

上机测验

- 1．设计函数 判断一个字符串是否为回文串（回文串是指从正向和反向两个方向读字符串都一样，例如 “Was it a rat I saw” 去掉空格后就是一个回文串，不考虑大小写），main 函数输入不带空格的字符串，输出 Y 或 N。

```
if(isPalindromeStr(a))
```

```
bool isPalindromeStr(const char str[])
{
    unsigned int length = strlen(str);
    for(int i=0, j=length-1; i < j; ++i, --j)
        if(str[i] != str[j])
            return false;
    return true;
}
```

回文串递归解法

- Was it a car or a cat I saw
- wasitacaroracatisaw

```
if(isPalindromeStr(a, 0, strlen(a)-1))
```

```
bool isPalindromeStr(const char str[], int i, int j)
{
     if(i >= j) return true;
     else
          if(str[i] != str[j])
               return false;
          else
               return isPalindromeStr(str, ++i, --j);
}
```

如果是一个整数呢?

- 转成字符串

```
bool is_plalindrome(unsigned int num)
{     char str[20], len=0;
      while (num != 0)
      {     str[len] = num % 10;
            num /= 10;
            ++len;
      }     //逆序
      for (int j = 0; j <= len/2; ++j)
      {     if (str[j] != str[len-j-1])
                  return false;
      }
      return true;
}
```

不转成字符串

```
bool isPla(int n)
{
      int m = 0;
      int temp = n;
      while(temp > 0)
      {
            m = m*10 + temp%10; // m 为 n 的逆序
            temp /= 10;
      }
      return m==n;
}
```

- **2．此前小黄车遇到了来自大量用户的押金退还请求，而小黄车的财务状况无法立即满足所有用户要求，故这些用户被要求进入等待队列排队等候。如果你是小黄车的 CTO 大佬，想在用户排队界面设计一个广告系统，用户点击了广告，就可以根据一定规则向前插队。这样既可以让用户竞争插队，也能因为广告收入解救公司的财务危机。请你先基于链表设计一个程序完成广告点击量最大的用户的插队操作，若有最大点击量相等的用户只让原始序号在先的用户插队，其他用户顺序不变。比如七个用户及点击量分别是：**

A B C D E F G

4 3 0 5 6 7 5

则他们的新序是：

F A B C D E G

7 4 3 0 5 6 5

```
struct Node
{
    char no;   //用户序号，自动按英文大写字母生成
    int hit;   //广告点击量，创建时输入
    Node *next;
};
```

```
int main( )
{
      Node *head = AppCreate( );
      PrintList(head);
      head = InsertKeyAhead(head, MaxHit(head));
      PrintList(head);
      DeleteList(head);
      return 0;
}
```

```
int MaxHit(Node *head)
{     int MaxHit = -1;
      char MaxNo = '\0';
      while(head)
      {     if(head -> hit > MaxHit)
            {     MaxHit = head -> hit;
                  MaxNo = head -> no;
            }
            head = head -> next;
      }
        return MaxNo;
```

```
Node *InsertKeyAhead(Node *h, char key)
{
    if(!h) return NULL;
    Node *current = h;
    Node *previous = NULL;
    while(current != NULL && current -> no != key )
    {    previous = current;
         current = current -> next;
    }//找key
    if(current != h)
    {    //if(current)
         previous -> next = current -> next;//删除key

         current -> next = h;
         h = current;                          //头部插入
    }

    return h;
}
```

```
Node *AppCreate( )
{    Node *head = NULL, *tail = NULL;
     int i;
     char ch = 'A';
     cin >> i;
     while( i != -1 )
     {
             Node *p = (Node *)malloc(sizeof(Node));
             p -> no = ch++;
             p -> hit = i;
             p -> next = NULL;
             if(head == NULL)
                    head = p;
             else
                    tail -> next = p;
             tail = p;
             cin >> i;
     }
     return head;
 }
```

```
void PrintList(Node *head)
{
	Node *h = head;
	while(head)
	{
		printf("%c ", head -> no);
		head = head -> next;
	}
	printf("\n");

	while(h)
	{
		printf("%d ", h -> hit);
		h = h -> next;
	}
	printf("\n");
}
```

第16-17周训练任务

- 1．现有1元、2元和5元的货币（数量不限），请设计程序计算购买价值为n元的物品共有多少种支付方式（要求输出每一种支付方式）。

穷举

```
#include <stdio.h>
int main( )
{
	int n = 0, count = 0;
	scanf("%d", &n);
	for(int i=0; i <= n; ++i)
	{
		for(int j=0; j <= n/2; ++j)
		{
			for(int k=0; k <= n/5; ++k)
			{
				if(i + j*2 + k*5 == n)
				{
					count++;
				}
			}
		}
	}
	printf("一共有 %d 种支付方式 \n", count);
	return 0;
}
```

2.从一副扑克牌（大小王各一张，A、2-10、J、Q、K各四张）中随机抽五张牌，判断是不是一个五连顺，即判断这五张牌是不是连续的。用整数代表扑克牌，1代表A，11代表J，12代表Q，13代表K。大、小王均用0代表，且可以看成任意牌。编程求解这个问题，输入一个长度为 5 的无序整型数组，输出yes/no.

分类

预处理

智力、经验

```
bool sq(int card[], int n)
{
    int count = 0, sum = 0;
    for( int i = 0; i < n; ++i)                    // 有几个0
        if(!card[i]) ++count;
    for( int i = count; i < n; ++i)                // 有无重复牌
        if(card[i]==card[i+1]) return false;
    for( int i = 0; i < n-1; ++i)
    {
        if(card[i])
            sum += card[i+1]-card[i]-1;            // 跨度和
    }
    if(sum > count)                                // 能否填平
        return false;
    else
        return true;
}
void sort(int card[], int n){ //排序函数略 }
```

不正确的做法，但貌似很对的亚子

注意考虑次序问题，正反两方面，不能浅尝辄止

```
//从大到小排序代码略
//注意0都在最后
bool flag = true;
for (int i=0; i < 4; i++)
{
      if (res[i] == 0 || res[i+1] == 0)
            break;
      if (res[i] - res[i + 1] != 1)
      {
            flag = false;
            break;
      }
}
```

当前面落差不为1，且没有足够的0填补的时候，程序恰好能够给出正确结果

当前面落差不为1，且**有**足够的0填补的时候，比如对于 6 4 8 7 0 或 5 0 7 0 8 则程序无法给出正确结果

3. 用包括main函数在内的多个函数，实现小数到分数的转换：向字符数组输入一个大于0的有限纯小数，输出其最简分数（例如，输入0.4，输出2/5；输入0.0125，输出1/80。小数部分不超过8位），用结构类型表示分数。转换函数原型为“Fraction decimalTofraction(char []);”。

```
int main()
{
    char str[10];
    cin >> str;
    //cout << strToNum(str) << endl;
    //cout << Pow10(str) << endl;
    Fraction f = decimalTofraction(str);
    cout << f.numerator << "/" << f.denominator << endl;
    return 0;
}
```

4/10 125/10000
分子分母分别考虑

```
int strToNum(char *str)
{
    int n=0, len = strlen(str)-2;
    str += 2;
    while(len > 0)
    {
            n = n*10 + (*str - '0');
            ++str;
            --len;
    }
    return n;
}
```

```
int Pow10(char *str)
{
    int d=1, len = strlen(str)-2; //几位小数

    while(len > 0)
    {
            d *= 10;
            --len;
    }
    return d;
}
```

```
int Gcd(int m, int n)
{
      while(n != 0)
      {
            int r = n;
            n = m % n;
            m = r;
      }
      return m;
}
```

```
Fraction decimalTofraction(char str[])
{
        int n = strToNum(str);
        int d = Pow10(str);
        Fraction f = {n, d};
        int g = Gcd(n, d);

        f.numerator /= g;
        f.denominator /= g;
        return f;
}
```

同学的做法

```
struct fenshu
{
	int fenzi;
	int fenmu;
};
int main()
{
	char str[11];
	cin.getline(str, 11);
	fenshu fs;
	fs.fenzi = 0, fs.fenmu = 1;
	for (int i = 2; str[i] != '\0'; ++i)
	{
		fs.fenzi = 10 * fs.fenzi + str[i] - '0';
		fs.fenmu = fs.fenmu * 10;
	}
	zuijian(fs.fenzi, fs.fenmu);
	cout << fs.fenzi << "/" << fs.fenmu << endl;
	return 0;
}
```

```
void zuijian(int &fz, int &fm)
{
	for (int i = 2; i <= fz; ++i)
	if (fz%i == 0 && fm%i == 0)
	{
		fz = fz / i;
		fm = fm / i;
		--i;
	}
}
```

4. 输入一个英文句子，识别句子中的不同单词，并按字母顺序输出。 注意：拼写相同大小写不同的单词应视为相同的单词，输出统一为小写的单词。要求使用链表存储输入和输出的单词。

```
#define N 50
#include <iostream>
using namespace std;
struct Node
{
    char word[N];
    Node *next;
};
```

```
int main()
{
    Node *head = Input();
    head = Sort(head);
    Print(head);
    Delete(head);
    return 0;
}
```

```
int strCmp(const char *src1, const char *src2)
{    while(*src1 == *src2)
     {    if(*src1 == '\0')
               return 0;
          ++src1, ++src2;
     }
     return *src1 - *src2;
}
```

```
Node *Input()
{     char ch;
      Node *head = NULL, *tail = head;
      do
      {     Node *p = new Node;
            p->next = NULL;
            cin >> p->word;
            if(head == NULL)
            {     head = p;
                  tail = head;
            }
            else
            {     tail->next = p;
                  tail = tail->next;
            }
      } while(ch != '\n');
      return head;
}
```

```
int i = 0;
```

```
ch = getchar();
while(ch != '\n' && ch != ' ')
{
      if(ch >= 'A'&&ch <= 'Z')
            ch += 'a' - 'A';
      p->word[i] = ch;
      ++i;
      ch = getchar();
}
p->word[i] = '\0';
```

参见课件链表插入法排序例子程序

- 只需修改:

```
if(p->data < head->data)
if(strCmp(p->word, head->word) < 0 )

if(p->data < cur->data)
if(strCmp(p->word, cur->word) < 0)
```

```
void Print(Node *head)
{
    while(head != NULL)
    {
        while(head->next!=NULL
                    && !strCmp(head->word, head->next->word))
            head = head->next;
        cout << head->word << ' ';
        head = head->next;
    }
    return;
}

void Delete(Node* head)
{
    …
}
```

或者删除相邻的重复节点，再输出

或者新建立一个链表，存入删除相邻的重复节点之后的链表，再输出

5. （选做）编写函数，实现单向链表的反转，即将链表的前后关系颠倒，头节点成为尾节点，尾节点成为头节点，第二个节点成为倒数第二个节点，以此类推。

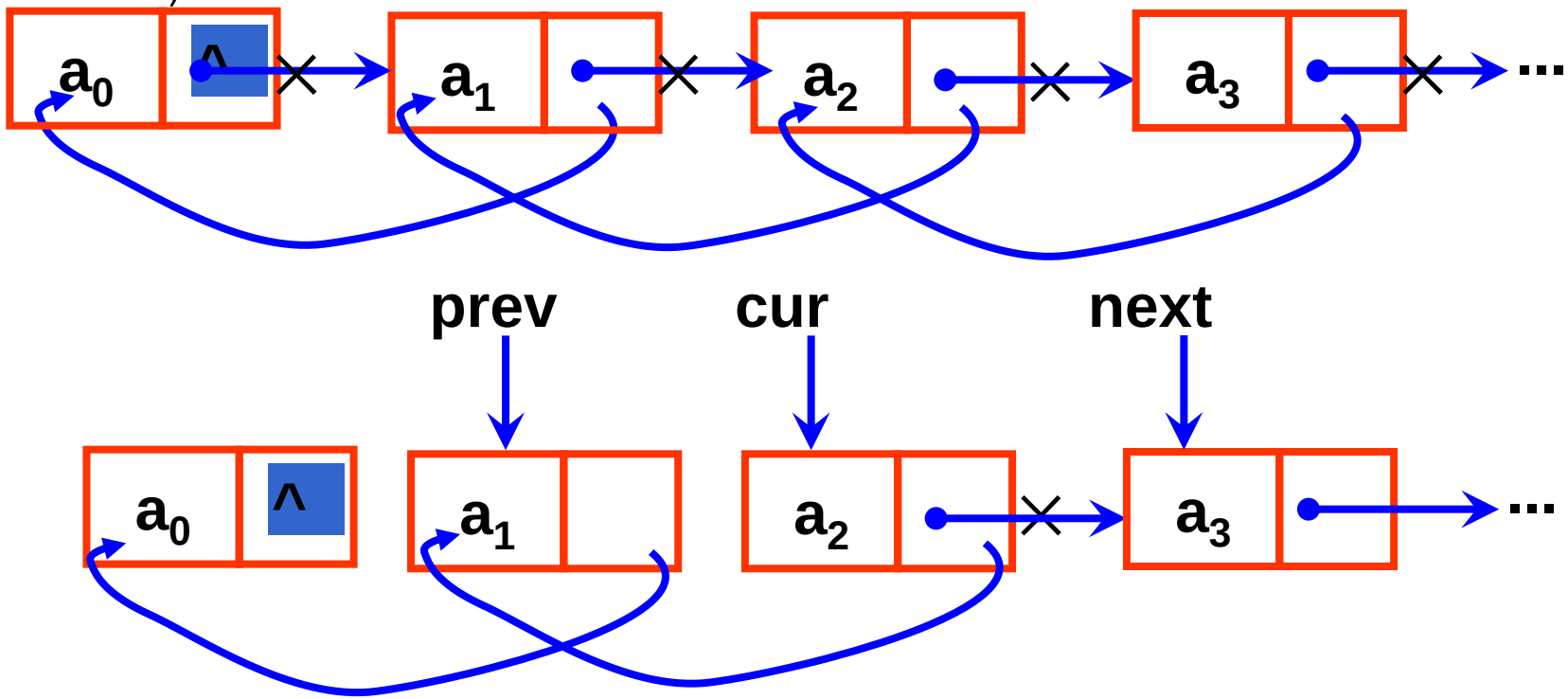
链表的反转**

- 链表的反转指的是将链表的前后关系颠倒，头节点变成尾节点，尾节点变成头节点。

- 解决这一问题的思路可以是：依次将每个当前节点的前一个节点链接在当前节点的后面，不过，事先要将当前节点、前一个节点、下一个节点的地址存于三个指针变量中，以免被覆盖，其中头节点的前一个节点和尾节点的下一个节点都看作空地址。

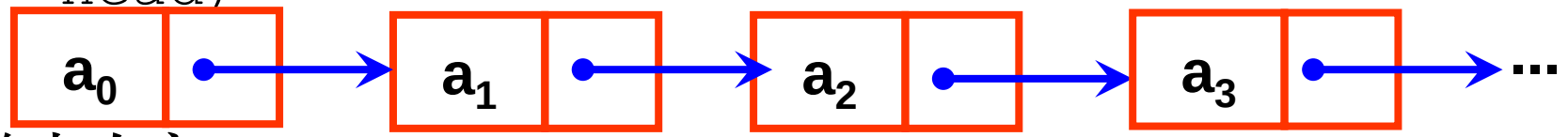
其实，用双向链表，易于实现链表的反转。

```
Node * Reverse(Node *head)
{
  Node *prev = NULL;
  Node *cur = NULL;
  Node *next = head;
  while(next != NULL)
  {
     prev = cur;
     cur = next;
     next = cur -> next;
     cur -> next = prev;
  }
  return cur;
}
```



用递归函数实现链表的反转**

- 基本思路：丢掉head节点，反转其余节点，逐渐缩小规模。




```
Node *RecReverse(Node *head)
{
  if(!head)      //整个链表为空
     return NULL;

  Node *temp = RecReverse(head -> next);//反转小规模链表
  if(!temp)      //小规模链表为空
     return head;

  head -> next  ->  next = head;     //在小规模链表尾部链接一个节点
  head -> next = NULL;     //小规模链表的尾部置空
  return temp; //返回小规模链表首节点的地址
}
```

6. （选做）用文件存储学生信息（学号 姓名 微积分测验成绩），输入一个学号，查询学生的微积分测验成绩，并输出。（略）

加法更容易

7.（选做） 设计程序，实现两个超长正整数的减法（思考设计全过程）。

```
void Reverse( char* acString )
{
      int iLength = strlen(acString);
      for( int i = 0; i < iLength/2; ++i )
      {
            swap( acString[i], acString[iLength-i-1] );
      }
}
```

```
23456
- 345
```

```
23456
- 345
```

```
int main()
{
    //A-B

    //初始化
    char* acA = new char[1024];
    memset( acA, 0, 1024*sizeof(char) );   //初始化
    char* acB = new char[1024];
    memset( acB, 0, 1024*sizeof(char) );   //初始化
    cin >> acA >> acB;
    char acResult[1024] = { 0 };
    int iLengthA = strlen(acA);
    int iLengthB = strlen(acB);
```

……

```
//ensure A>B
bool bNegative = false;
if(iLengthA < iLengthB
          || (iLengthA == iLengthB)&&(strcmp(acA, acB)<0))
{
     bNegative = true;
     swap( iLengthA, iLengthB );
     swap( acA, acB );
}

//reverse string
Reverse( acA );
Reverse( acB );
```

……

```
//minus per digit 从低位开始减
for( int i = 0; i < iLengthB; ++i )
{
     acResult[i] = acA[i] - acB[i];
}

for( int i = iLengthB; i < iLengthA; ++i )
{
     acResult[i] = acA[i] - '0';
}
```

```
23456
- 345
```

```
23456
- 345
```

……

```
//handle negative digits 借位
for( int i = 0; i < iLengthA; ++i )
{
     if( acResult[i] < 0 )
     {
          acResult[i] += 10;
          --acResult[i+1];
     }
}
```

……

```
//remove redundant 0
int iLength = iLengthA;
for( ; iLength > 0; --iLength )
{
     if( acResult[iLength-1] != 0 )
     {
          break;
     }
}

//special case: the answer is 0
if( iLength == 0 )
     iLength = 1;
```

……

```
    //transfer integer to character
    for( int i = 0; i < iLength; ++i )
         acResult[i] += '0';

    //add '-' if negative
    if( bNegative )
    {
         acResult[iLength] = '-';
         acResult[iLength+1] = '\0';
    }
    else
         acResult[iLength] = '\0';
    //reverse back
    //输出、delete[] acA;      delete[] acB;
    return 0;
}
```

祝大家期末考试取得好成绩！

Thanks！