《传智播客 C语言就业班》 第七讲 结业考试

```
/*
1 编写一个业务函数，实现字符串（前后各有三个空格，单词前后也均有空格）
   "   i am student, you are teacher   " ,
   各个单词首字符大写，结果如下"   i am student,  you are teacher   " ,
   要求1：实现所有接口 70
   要求2：写出测试程序 30
*/

/****************************************************************/
/**************************** 方法1 ****************************/
/*****************一级指针做输入，主调函数分配内存****************/
/****************************************************************/
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

int GetString(const char *str1 /*in*/, char *str2 /*in*/)
{
   int ret = 0;
   char *p1 = NULL, *p2 = NULL;
   int len = 0, i = 0;
   if (str1 == NULL || str2==NULL )
   {
      ret = -1;
      return ret;
   }
   len = strlen(str1);
   //
   strcpy(str2, str1);
   //初始化循环环境
   p1 = str2 +1 ;
   p2 = str2;
   for (i=0; i<len && (*p1!= NULL) ; i++)
   {
      if  (isalpha(*p1) && isspace(*p2))
      {
         *p1 = *p1 - 32;
      }
      //打造循环条件
      p2 = p1;
      p1 ++;
   }
   return ret;
}
void main01()
{
   int ret = 0;
   const char *str1 = "     i am student, you are teacher   ";
   char buf[1024] = {0};
   ret = GetString(str1 /*in*/, buf /*in*/);
   if (ret != 0)
   {
      printf("func GetString() err:%d \n", ret);
      return ret;
   }
   printf("buf:%s \n", buf);
   system("pause");
}
```

```
/*****************************************************************/
/***************************** 方法2 ****************************/
/*****************二级指针做输入，被调函数分配内存****************/
/*****************************************************************/
int GetString_Adv(const char *str1/*in*/, char **str2 /*out*/)
{
    int ret = 0;
    char *p1 = NULL, *p2 = NULL;
    char *ptmp = NULL;
    int len = 0, i = 0;
    if (str1 == NULL || str2==NULL )
    {
        ret = -1;
        return ret;
    }
    len = strlen(str1);
    ptmp = (char *)malloc((len+1) * sizeof(char));
    if (ptmp == NULL)
    {
        ret = -2;
        return ret;
    }
    memset(ptmp, 0, (len+1) * sizeof(char));
    //
    strcpy(ptmp, str1);
    //初始化循环环境
    p1 = ptmp +1 ;
    p2 = ptmp;
    for (i=0; i<len && (*p1!= NULL) ; i++)
    {
        if  (isalpha(*p1) && isspace(*p2))
        {
            *p1 = *p1 - 32;
        }
        //打造循环条件
        p2 = p1;
        p1 ++;
    }
    *str2 = ptmp;  //str2是实参的地址 间接赋值修改实参，
                  //让实参指向新分配的内存空间
    return ret;
}
int GetString_Adv_Free1(char *str2)
{
    if (str2 == NULL)
    {
        return -1;
    }
    free(str2);
    str2 = NULL; //垃圾语句，建议按照Free2函数来写
}
int GetString_Adv_Free2(char **str2)
{
    char *tmp = NULL;
    if (str2 == NULL)
    {
        return -2;
    }
    tmp = *str2;
    if (tmp != NULL)
    {
        free(tmp);
        *str2 = NULL;
    }
```

```c
}
void main02()
{
    int ret = 0;
    const char *str1 = "   i am student, you are teacher   ";
    //char buf[1024] = {0};
    char *pbuf = NULL;
    ret = GetString_Adv(str1 /*in*/, &pbuf /*in*/);
    if (ret != 0)
    {
        printf("func GetString() err:%d \n", ret);
        return ret;
    }
    printf("pbuf:%s \n", pbuf);
    GetString_Adv_Free2(&pbuf);
    system("pause");
}
```

```c
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
/*
2 编写一个业务函数，实现按行读取文件。
把内容按照第三种内存模型打包数据传出，把行数通过函数参数传出。
函数原型有两个，任意选择其一
要求1：请自己任意选择一个接口（函数），并实现功能；70分
要求2：编写测试用例。30分
要求3：自己编写内存释放函数
*/

//第1种写法：
char **readFile1(const char *pfilename/*in*/, int *lineNum/*in out*/)
{
    int     rv = 0;
    FILE    *fp = NULL;
    char     lineBuf[1024*4];

    char    **pTmp = NULL;
    char *p = NULL;
    int tmpLine = 0, strLine = 0, i = 0;
    if (pfilename==NULL || lineNum==NULL )
    {
        rv = -1;
        printf("readFile1() err. param err \n");
        goto End;
    }
    fp = fopen(pfilename, "r");
    if (fp == NULL)
    {
        rv = -2;
        printf("fopen() err. \n");
        goto End;
    }

    //第一遍 读取文件有多少行
    while (!feof(fp))
    {
        //读每一行
        memset(lineBuf, 0, sizeof(lineBuf));
        p = fgets(lineBuf, 1024*4, fp);
        if (p == NULL)
        {
            break;
        }
        else
        {
            tmpLine ++;
        }
    }
    pTmp = (char **)malloc(tmpLine * sizeof(char *));
    if (pTmp == NULL)
    {
        rv = -2;
        printf("malloc() err. \n");
        goto End;
    }
    //让文件指针指向文件的开头，目的：第二次从头检索
    fseek(fp, 0L, SEEK_SET);

    //
    i = 0;
    while (!feof(fp))
    {
        //读每一行
        memset(lineBuf, 0, sizeof(lineBuf));
        p = fgets(lineBuf, 1024*4, fp);
        if (p == NULL)
        {
            break;
        }
        strLine = strlen(lineBuf);
        pTmp[i] = (char *)malloc((strLine + 1) * sizeof(char));
        if (pTmp[i] == NULL)
        {
            rv = -3;
            printf("malloc() err. \n");
            goto End;
        }
        strcpy(pTmp[i], lineBuf);
        i++;
    }

End:
    if (fp != NULL)
    {
        fclose(fp);
    }
    //赋值
    *lineNum = tmpLine;
    return pTmp;
}

void FreeMypp(char **p, int linenum)
{
    int i = 0;
    if (p == NULL)
    {
        return NULL;
    }
    for (i=0; i<linenum; i++)
    {
        if (p[i] != NULL)
        {
            free(p[i]) ;
        }
    }
    free(p);
    return ;
```

```
}

void main21()
{
    char            ** mypp = NULL;
    const char      *pfilename = "c:/1.txt";
    int             lineNum = 0, i = 0;
    mypp = readFile1(pfilename/*in*/, &lineNum/*in out*/);
    if (mypp == NULL)
    {
        return ;
    }
    for (i=0; i<lineNum; i++)
    {
        printf("%s\n", mypp[i]);
    }
    system("pause");
}

//第2种写法：
int readFile1_Adv(const char *pfilename/*in*/,char ***myfileP, int *lineNum/*in out*/)
{
    int     rv = 0;
    FILE    *fp = NULL;
    char    lineBuf[1024*4];
    char    **pTmp = NULL;
    char *p = NULL;
    int tmpLine = 0, strLine = 0, i = 0;
    if (pfilename==NULL || lineNum==NULL || myfileP==NULL)
    {
        rv = -1;
        printf("readFile1() err. param err \n");
        goto End;
    }
    fp = fopen(pfilename, "r");
    if (fp == NULL)
    {
        rv = -2;
        printf("fopen() err. \n");
        goto End;
    }
    //第一遍 读取文件有多少行
    while (!feof(fp))
    {
        //读每一行之前先初始化lineBuf
        memset(lineBuf, 0, sizeof(lineBuf));
      //一般每行不超过4个字节，存在lineBuf字符数组中；
        p = fgets(lineBuf, 1024*4, fp);
        if (p == NULL)
        {
            break;
        }
        else
        {
            tmpLine ++;
        }
    }
    pTmp = (char **)malloc(tmpLine * sizeof(char *));
    if (pTmp == NULL)
    {
        rv = -2;
        printf("malloc() err. \n");
        goto End;
    }
    //让文件指针指向文件的开头，目的：第二次从头检索
    fseek(fp, 0L, SEEK_SET);
    //
    i = 0;
    while (!feof(fp))
    {
        //读每一行
        memset(lineBuf, 0, sizeof(lineBuf));
        p = fgets(lineBuf, 1024*4, fp);
        if (p == NULL)
        {
            break;
        }
        strLine = strlen(lineBuf);
        pTmp[i] = (char *)malloc((strLine + 1) * sizeof(char));
        if (pTmp[i] == NULL)
        {
            rv = -3;
            printf("malloc() err. \n");
            goto End;
        }
        strcpy(pTmp[i], lineBuf);
        i++;
    }

End:
    if (fp != NULL)
    {
        fclose(fp);
    }
    //赋值
    *lineNum = tmpLine;
    *myfileP = pTmp;
    return rv;
}

void main22()
{
    int          ret = 0;
    char             ** mypp = NULL;
    const char       *pfilename = "c:/1.txt";
    int          lineNum = 0, i = 0;
    ret = readFile1_Adv(pfilename/*in*/, &mypp, &lineNum/*in out*/);
    if (ret != 0)
    {
        return ;
    }
    for (i=0; i<lineNum; i++)
    {
        printf("%s\n", mypp[i]);
    }
    FreeMypp(mypp, lineNum);
    system("pause");
}

//第3种写法：不传行数也可以；读取或释放时，有自我结束的能力，见后面
int readFile1_Adv2(const char *pfilename/*in*/,char ***myfileP/*in out*/)
{
    int     rv = 0;
    FILE    *fp = NULL;
    char     lineBuf[1024*4];
    char    **pTmp = NULL;
    char *p = NULL;
    int tmpLine = 0, strLine = 0, i = 0;
    if (pfilename==NULL || myfileP==NULL)
    {
        rv = -1;
        printf("readFile1() err. param err \n");
        goto End;
    }
    fp = fopen(pfilename, "r");
    if (fp == NULL)
    {
        rv = -2;
        printf("fopen() err. \n");
        goto End;
    }
    //第一遍 读取文件有多少行
    while (!feof(fp))
    {
        //读每一行
        memset(lineBuf, 0, sizeof(lineBuf));
        p = fgets(lineBuf, 1024*4, fp);
        if (p == NULL)
        {
            break;
        }
        else
        {
            tmpLine ++;
        }
    }
    pTmp = (char **)malloc((tmpLine+1) * sizeof(char *));
    if (pTmp == NULL)
    {
        rv = -2;
        printf("malloc() err. \n");
        goto End;
    }
    memset(pTmp, 0, (tmpLine+1) * sizeof(char *));
    //让文件指针指向文件的开头，目的：第二次从头检索
    fseek(fp, 0L, SEEK_SET);
    //
    i = 0;
    while (!feof(fp))
    {
        //读每一行
        memset(lineBuf, 0, sizeof(lineBuf));
        p = fgets(lineBuf, 1024*4, fp);
        if (p == NULL)
        {
            break;
        }
        strLine = strlen(lineBuf);
        pTmp[i] = (char *)malloc((strLine + 1) * sizeof(char));
        if (pTmp[i] == NULL)
        {
            rv = -3;
            printf("malloc() err. \n");
            goto End;
        }
        strcpy(pTmp[i], lineBuf);
        i++;
    }

End:
    if (fp != NULL)
    {
        fclose(fp);
    }
    //赋值
    //*lineNum = tmpLine;
    *myfileP = pTmp;
    return rv;
}

void FreeMypp3(char **p)
{
    int i = 0;
    if (p == NULL)
    {
        return NULL;
    }
    for (i=0; p[i]!=NULL; i++)
    {
        if (p[i] != NULL)
        {
            free(p[i]) ;
        }
    }
    free(p);
    return ;
}

void main23()
{
    int             ret = 0;
    char                ** mypp = NULL;
    const char      *pfilename = "c:/1.txt";
    int             lineNum = 0, i = 0;
    ret = readFile1_Adv3(pfilename/*in*/, &mypp/*in out*/);
    if (ret != 0)
    {
        return ;
    }
    for (i=0; mypp[i]!=NULL; i++)
    {
        printf("%s\n", mypp[i]);
    }
    FreeMypp3(mypp);
    system("pause");
}
```

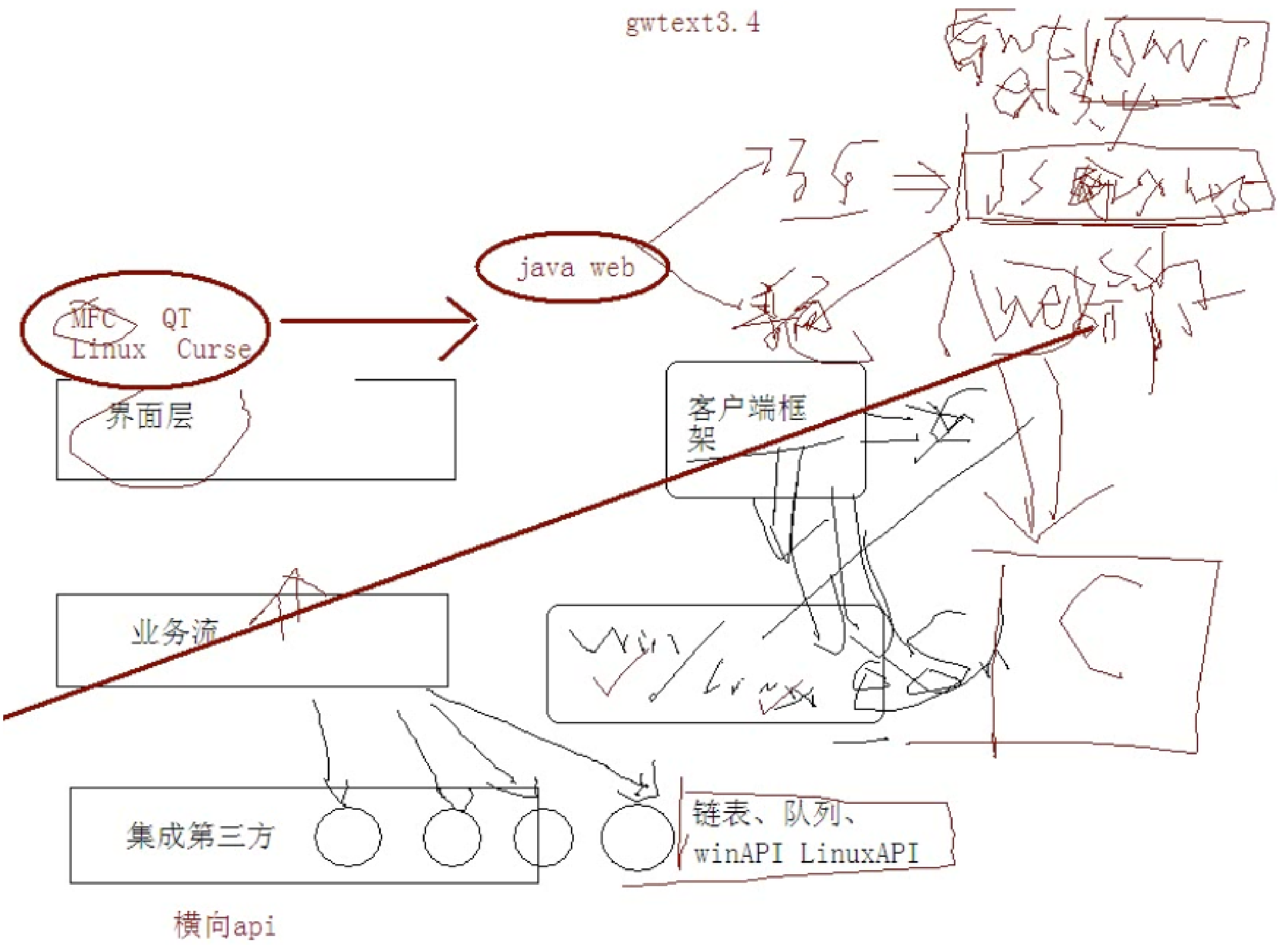
```
/*3 链表如下
typedef struct _LinkList
{
    int data;
    struct _LinkList*next;
} LinkList;
有如下结点数据域  1 2 3 4 5 6 7 8 12 19。。。。
要求1：创建链表
要求2：删除结点值为偶数的结点 ；70分
要求3：编写测试用例  30分
*/

#include "stdio.h"
#include "stdlib.h"
#include "string.h"

typedef struct Node
{
    int data;
    struct Node *next;
}SLIST;
```

```
//编写函数SList_Creat，建立带有头结点的单向链表。循环创建结点，
//结点数据域中的数值从键盘输入，以-1作为输入结束标志。链表的头结点地址由函数值返回。
SLIST *SList_Creat()
{
   SLIST *pHead = NULL, *pM =NULL, *pCur = NULL;
   int data = 0;
   //1 创建头结点并初始化
   pHead = (SLIST *)malloc(sizeof(SLIST));
   if (pHead == NULL)
   {
      return NULL;
   }
   pHead->data = 0;
   pHead->next = NULL;
   //2 从键盘输入数据，创建业务结点
   printf("\nplease enter the data of node(-1:quit) ");
   scanf("%d", &data);
   //3 循环创建
   //初始化当前节点，指向头结点
   pCur = pHead;
   while(data != -1)
   {
      //新建业务结点 并初始化
      //1 不断的malloc 新的业务节点 ===PM
      pM = (SLIST *)malloc(sizeof(SLIST));
      if (pM == NULL)
      {
         SList_Destory(pHead); //
         return NULL;
      }
      pM->data = data;
      pM->next = NULL;
      //2、让pM节点入链表
      pCur->next = pM;
      //3 pM节点变成当前节点
      pCur = pM; //pCur = pCur->next;
      //从键盘输入数据，创建业务结点
      printf("\nplease enter the data of node(-1:quit) ");
      scanf("%d", &data);
   }
   return pHead;
}

int SList_Print(SLIST *pHead)
{
   SLIST *p = NULL;
   if (pHead == NULL)
   {
      return -1;
   }
   p = pHead->next;
   printf("\nBegin ");
   while(p)
   {
      printf("%d ", p->data);
      p  = p->next;
   }
   printf(" End");
   return 0;
}

int SList_Destory(SLIST *pHead)
{
   SLIST *p = NULL, *tmp = NULL;
```

```c
    if (pHead == NULL)
    {
        return -1;
    }
    p = pHead;
    while(p)
    {
        //缓存下一个结点位置
        tmp = p->next;
        free(p);//删除当前节点
        p = tmp; //节点指针后移
    }
    return 0;
}

//删除节点值为偶数的
int SList_NodeSpecialDel(SLIST *pHead)
{
    SLIST *pCur = NULL, *pPre = NULL;
    //准备pCur Pre环境
    pPre = pHead;
    pCur = pHead->next;
    while (pCur)
    {
        if (pCur->data %2 == 0)
        {
            //插入操作
            break;
        }
        pPre = pCur; //让前驱结点后移
        pCur = pCur->next; //让当前结点后移
    }
    if (pCur == NULL)
    {
        printf("没有找到要删除的结点\n");
        return 0;
    }
    //从链表上删除结点
    pPre->next =   pCur->next;
    //释放内存
    free(pCur);
    return 0;
}

int SList_NodeDelOueve(SLIST *pHead)
{
    SLIST *p = NULL;
    if (pHead == NULL)
    {
        return -1;
    }
    p = pHead->next;
    while (p)
    {
         SList_NodeSpecialDel(p);
        p = p->next;
    }
    return 0;
}

void main()
{
    int ret = 0;
    SLIST *pHead = NULL;
    pHead = SList_Creat();
    ret = SList_Print(pHead);
    if (ret != 0)
    {
        printf("func SList_Print() err:%d\n", ret);
        return ;
    }
    //删除结点值为偶数的结点
    ret = SList_NodeDelOueve(pHead);
    if (ret != 0)
    {
        printf("func SList_NodeDelOueve(pHead)() err:%d\n", ret);
        return ;
    }
    ret = SList_Print(pHead);
    if (ret != 0)
    {
        printf("func SList_Print() err:%d\n", ret);
        return ;
    }
    ret = SList_Destory(pHead);
    if (ret != 0)
    {
        printf("func SList_Destory() err:%d\n", ret);
        return ;
    }
    system("pause");
}
```

```
/*
4、从键盘中输入一个不超过40个字符的字符串，再输入3个位数（每次删除一个字符），
删除对应位数的字符，然后输出删除指定字符后的字符串。
    输入：hellokityManGood
    3  6   9
    helokityManGood
    heloktyManGood
    heloktyMnGood
*/
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
int delAlpa(char *p, int pos)
{
    int len = 0;
    int i = 0;
    if (p == NULL)
    {
        return -1;
    }
    len = strlen(p);
    if (pos >= len)
    {
        return -2;
    }
    //pos位置从0开始
    for (i=pos; i<len; i++)
    {
        p[i] = p[i+1];
    }
    p[len-1] = '\0';
    return 0;
}
void main()
{
    int ret = 0;
    char str[] = "hellokityManGood";
    ret = delAlpa(str, 3);
    if (ret != 0)
    {
        printf("func delAlpa() err:%d \n",ret);
        return ret;
    }
    printf("str:%s \n", str);
    ret = delAlpa(str, 6);
    if (ret != 0)
    {
        printf("func delAlpa() err:%d \n",ret);
        return ret;
    }
    printf("str:%s \n", str);
    ret = delAlpa(str, 9);
    if (ret != 0)
    {
        printf("func delAlpa() err:%d \n",ret);
        return ret;
    }
    printf("str:%s \n", str);
    system("pause");
}
```

软件开发方向选择


gwtext3.4
java web
MFC QT
Linux Curse
界面层
客户端框架
业务流
集成第三方
链表、队列、
winAPI LinuxAPI
横向api