Lab 7 Part II

Lab 7 Part II 目标是对指针深入理解。

Problem 7. 指针最重要的作用是灵活地对内存进行解析，这也是学好指针的关键所在。定义一个 32 字节大小的字符数组 char S[32]，内存布局如下（表格第二行表示第 17-32 个字符）：

‘a’	‘b’	‘c’	‘ ‘	‘d’	‘e’	‘f’	‘\0’	‘g’	‘h’	‘i’	‘ ‘	‘j’	‘k’	‘l’	‘\0’
‘m’	‘n’	‘o’	‘ ‘	‘p’	‘q’	‘\0’	‘r’	‘s’	‘t’	‘u’	‘ ‘	‘v’	‘\0’	‘x’	‘y’

编程验证以下程序的输出结果，并解释原因：

(1)

```
cout<<S<<endl;
cout<<S+1<<endl;
cout<<S+8<<endl;

cout<<&S[0]<<endl;
cout<<&S[1]<<endl;
cout<<&S[8]<<endl;
```

(2)

```
int *p = (int *)S; //将内存解释为若干个 int 类型变量
cout<<p<<endl;
cout<<*p<<endl;
cout<<p[0]<<endl;
cout<<*(p+1)<<endl;
cout<<p[1]<<endl;
cout<<*p + 1<<endl;
```

(3)

```
float *p = (float *)S; //将内存解释为若干个 float 类型变量
cout<<p<<endl;
cout<<*p<<endl;
cout<<p[0]<<endl;
cout<<*(p+1)<<endl;
cout<<p[1]<<end;
cout<<*p + 1<<endl;
```

(4)

```
double *p = (double *)S; //将内存解释为若干个 double 类型变量
cout<<p<<endl;
cout<<*p<<endl;
```

```
cout<<*(p+1)<<endl;
cout<<*p + 1<<endl;
```

(5)

```
unsigned int (*p) [2] = (unsigned int (*)[2])S; //将内存解释为若干个 unsigned int [2]类型的数组
cout<<p<<endl;
cout<<*p<<endl;
cout<<*(p+1)<<endl;
cout<<**p<<endl;
cout<<*(*p+1)<<endl;
cout<<**(p+1)<<endl;
cout<<p[0]<<endl;
cout<<p[0][0]<<endl;
cout<<p[1][0]<<endl;
cout<<p[0][1]<<endl;
```

(6)

```
char (*p)[8] = (char (*)[8])S; //将内存解释为若干个 char [8]类型的数组
cout<<p<<endl;
cout<<*p<<endl;
cout<<*(p+1)<<endl;
```

(7)

```
unsigned short (*p) [2][2] = (unsigned short (*) [2][2])S; //将内存解释为若干个 unsigned short [2][2]类型的数组
cout<<p<<endl;
cout<<*p<<endl;
cout<<*(p+1)<<endl;
cout<<**p<<endl;
cout<<*(*p+1)<<endl;
cout<<**(p+1)<<endl;
cout<<***p<<endl;
cout<<***(p+1)<<endl;
cout<<**(*p + 1)<<endl;
cout<<*(**p + 1)<<endl;
cout<<p[0]<<endl;
cout<<p[0][0]<<endl;
cout<<p[1][0]<<endl;
cout<<p[0][1]<<endl;
cout<<p[0][0][0]<<endl;
cout<<p[0][1][0]<<endl;
```

Problem 8. 动态分配内存（**完成后熟记背诵下来**）。

(1) 从键盘输入整数 n（n 为变量），动态分配大小为 n 的 int 类型数组，从键盘输入 n 个整

数，排序后输出。

(2) 从键盘输入整数 n（n 为变量），动态分配大小为 n 的字符型数组，从键盘输入字符串（长度小于 n），将字符串反序输出。

(3) 从键盘输入整数 n 和 m（m 和 n 都是变量），表示一个矩阵的行和列。从键盘输入 n 行数据，每行包含 m 个整数，空格隔开，代表矩阵元素。使用动态分配内存存储该矩阵，并将矩阵转置后输出（提示：先动态生成一维指针数组，每个指针再赋值为动态一维数组）。

Problem 9. 复现 memcpy。

memcpy 的功能是将一段内存的内容复制到另一段内存，例如，将一个长度为 16 字节的 char 型数组的内容复制到一个有 4 个元素的 int 型数组。根据下面函数原型，实现 memcpy 函数。（**注意如果 src 和 dest 有 overlap 的情况！**）

```
void mymemcpy(void *src, void *dest, int size) {
//src 表示源内存地址，desti 表示目标内存地址，size 表示要拷贝的内存大小
}
```

Problem 10. **仔细阅读下面程序，读懂后，熟记背诵下来。**

```
#include <iostream>
#include <cstring>
using namespace std;

struct link{
    int  elem;
    struct link *next;
};

link * initLink();

link * insertElem(link * p,int elem,int add);

link * delElem(link * p,int add);

int selectElem(link * p,int elem);

link *amendElem(link * p,int add,int newElem);

void display(link *p);

int main() {
    link *p=initLink();
    display(p);
```

```
    p=insertElem(p, 5, 4);
    display(p);

    p=delElem(p, 3);
    display(p);

    int address=selectElem(p, 2);

    if (address==-1) {
        cout<<"没有该元素";
    }else{
        cout<<"元素 2 的位置为 "<<address;
    }

    cout<<"更改第 3 的位置的数据为 7:"<<endl;

    p=amendElem(p, 3, 7);
    display(p);

    return 0;
}

link * initLink(){
    link * p=(link*)new link;;
    link * temp=p;

    for (int i=1; i<5; i++) {
        link *a=(link*)new link;
        a->elem=i;
        a->next=NULL;
        temp->next=a;
        temp=temp->next;
    }

    return p;
}

link * insertElem(link * p,int elem,int add){
    link * temp=p;

    for (int i=1; i<add; i++) {
        if (temp==NULL) {
            cout<<"插入位置无效"<<endl;
```

```
            return p;
        }
        temp=temp->next;
    }

    link * c=(link*) new link;
    c->elem=elem;

    c->next=temp->next;
    temp->next=c;

    return  p;
}

link * delElem(link * p,int add){
    link * temp=p;

    for (int i=1; i<add; i++) {
        temp=temp->next;
    }

    link * del=temp->next;
    temp->next=temp->next->next;
    delete del;

    return p;
}

int selectElem(link * p,int elem){
    link * t=p;
    int i=1;

    while (t->next) {
        t=t->next;
        if (t->elem==elem) {
            return i;
        }
        i++;
    }

    return -1;
}

link *amendElem(link * p,int add,int newElem){
```

```
    link * temp=p;
    temp=temp->next;

    for (int i=1; i<add; i++) {
        temp=temp->next;
    }
    temp->elem=newElem;

    return p;
}

void display(link *p){
    link* temp=p;

    while (temp->next) {
        temp=temp->next;
        cout<<temp->elem;
    }
    cout<<endl;
}
```