**实验九 指针与数组**

**一、实验目的和要求**

编写程序，将输入的一个字符串进行加密和解密。加密时，每个字符依次反复加上“8734962”中的数字，如果范围超过ASCII码的032(空格)~122（‘z’），则进行模运算。解密与加密的顺序相反。编制加密和机密的函数，打印各个过程的结果。

**二、实验内容**

1. 实验准备
   1. 理论知识介绍
      1. 数据的输入：

要从用户依次读入int类型的变量a,n，可以使用如下语句:

cin >> a>>n;

* + 1. 数据的输出

要将变量的内容显示在显示器上，可以使用cout和流插入运算符。Cout还可以输出表达式的执行结果，比如:

cout << a << ‘+’ << b << ‘=’ << a+b << endl;

* + 1. 算术表达式

C++中，算术运算里有+（加法），-（减法），/（除法），%（取模），但是没有取平方算数运算。

* + 1. 数据类型 （整型）：

C++中一个整型变量可以储存一个整数，可以直接通过cin和cout输入和输出。

* + 1. 关系表达式

关系表达式用于比较两个值的大小。C++提供了6个关系运算符: <, <=, >, >=,

==, !=。关系运算符左结合。

* + 1. while语句

**while (表达式) {**

**需要重复执行的语句**

**}**

* + 1. switch 语句

**switch (控制表达式) {**

**case 常量表达式1: 语句1； break;**

**case 常量表达式2: 语句2； break;**

**…**

**case 常量表达式n: 语句n； break;**

**default: 语句n+1;**

**}**

* + 1. 函数

编写一个实现某个功能的函数称为函数定义。一旦定义了一个函数，在程序中就可以反复调用这个函数。函数的定义形式如下：

**类型名 函数名 (形式参数表)**

**{**

**变量定义部分**

**语句部分**

**}**

* + 1. 指针

所谓的指针变量就是保存另一个变量地址的变量。指针变量存在的意义在于提供间接访问，即从一个变量访问另一个变量的值，使变量访问更加灵活。

指针变量的定义如下：

**类型名 \*指针变量名**

1. 实验项目
   1. 分析
      1. 由于该程序要求我们提供加密和解密两个不同的功能，所以我们需要提供一个标识，输入不同的标识能产生不同的功能。
      2. 要有两个不同的功能，所以我们要设计两个不同的函数，一个用于加密，另一个用于解密。
      3. 通常函数不能改变实参的值，但是如果传入的是指针，就能通过指针改变实参的值。
   2. 方案
      1. 从用户读入需要进行操作的字符串和需要进行的操作。
      2. 利用switch语句进行相应的操作。
      3. 利用指针向加密解密函数传递字符串的指针参数从而改变字符串的值。
      4. 输出处理后的结果
   3. 源程序

#include <iostream>

using namespace std;

char \*encryption(char \*str, int key[]) {

char \*p = str;

int i = 0;

while (\*p != '\0') {

\*p = (\*p + key[i] - 32) % 91 + 32;

++p;

i = (i + 1) % 7;

}

return str;

}

char \*decryption(char \*str, int key[]) {

char \*p = str;

int i = 0;

while (\*p != '\0') {

\*p = (\*p - key[i] + 59) % 91 + 32;

++p;

i = (i + 1) % 7;

}

return str;

}

int main() {

int flag;

char text[100];

int key[] = { 8, 7, 3, 4, 9, 6, 2 };

cout << "Please enter the string: " << endl;

cin.getline(text, 100);

cout << "What do you want to do?" << endl;

cout << "[1] Encryption" << endl;

cout << "[2] Decryption" << endl;

cin >> flag;

switch (flag) {

case 1 :

cout << encryption(text, key);

break;

case 2 :

cout << decryption(text, key);

break;

default:

break;

}

return 0;

}

1. 测试数据运行结果
   1. Please enter the string:

SJTU

What do you want to do?

[1] Encryption

[2] Decryption

1

[QWY

* 1. Please enter the string:

[QWY

What do you want to do?

[1] Encryption

[2] Decryption

2

SJTU

* 1. Please enter the string:

Hello SJTU

What do you want to do?

[1] Encryption

[2] Decryption

1

Plopx&UR[X

* 1. Please enter the string:

Plopx&UR[X

What do you want to do?

[1] Encryption

[2] Decryption

2

Hello SJTU

1. 异常现象

本来这里的语句段是这样的

cout << "What do you want to do?" << endl;

cout << "[1] Encryption" << endl;

cout << "[2] Decryption" << endl;

cin >> flag;

cout << "Please enter the string: " << endl;

cin.getline(text, 100);

但是如果程序是这样的话，整个程序不知道什么原因并不会读入text这个字符串的量，而且程序也不会有任何的输出。

1. 分析

可能是因为cin.getline()这个函数具有特殊性，和cin这个操作有冲突，将其放置的在cin >> flag 之前就能产生效果，放置在它后面就没有效果。

1. 总结

因为cin.getline()这个函数的特殊性，所以我们必须要先把cin.getline()这个函数放置在cin这个操作之前。

**三、实验小结**

1. 完成情况

本来cin.getline()函数不能产生效果的问题不能解决，查阅了相关的资料之后发现由于cin的影响，输出流还存在一个换行符，所以如果要将cin.getline()放置于cin之后，我们就要添加这样一个语句：

cin.ignore();

这样的话换行符就能被忽略，cin.getline()才能产生效果。

1. 重点

利用指针作为函数的参数和返回值。

1. 难点

如何对每个字符进行想要的加密的效果。

1. 解决方法

通过模运算并加上适当的常数之后减去相应的常数得到结果。

1. 有待改进之处

可以在switch语句之外添加一个while循环的语句并在switch语句中添加一个退出的选项，将程序修改成一直运行的程序。

1. 收获、体会

指针为我们间接访问变量提供了便利，同时我们也能利用指针参数修改函数之外的变量。同时有些我们曾经接触过的函数其实我们并不熟悉，比如cin.getline()我们就不知道不能将其放置在另一个cin之后，所以很可能会出现异常。