**实验十 指针数组与函数数组**

1. **实验目的和要求**

利用随机数生成器开发一个洗牌和发牌的模拟程序。

一副扑克牌有52张牌，其中52张是正牌，另2张是副牌（大王和小王）。52张正牌又均分为13张一组，并以黑桃、红桃、梅花、方块四种花色表示各组，每组花色的牌包括从1~10（1通常表示为A）以及J、Q、K标示的13张牌，玩法千变万化。

你可以利用下面提供的数据结构：

const char \*suit[4] = { "Spades", "Hearts", "Clubs", "Diamonds" };表示扑克牌的四个花色。

const char \*face[13] = { "Ace", "Deuce", "Three", "Four", "Five", "Six",

"Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King" }; 表示每组花色的13张牌的名称。

int deck[4][13];表示一副要玩的牌。不考虑大王和小王。

**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. for语句

**for (i = 0; i < n; ++i)**

**需要重复执行的语句**

* + 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. 函数指针

指向函数的指针的定义为：

**返回类型 （\*指针变量） （形式参数列表）；**

指向函数的指针主要有两个用途：作为另一个函数的参数，以及用于实现菜单选择。

利用函数指针我们能将一个函数作为另一个函数的参数。

1. 实验项目
   1. 分析
      1. 由于我们要实现洗牌和发牌这两个功能，所以要设计两个函数分别实现这个功能。
      2. 同时为了灵活地利用这两个函数，我们可以再定义一个函数，将这两个函数作为这个函数的参数，这样就能比较方便地实现选择列表的功能。
   2. 方案
      1. 由用户输入一个标志变量
      2. 通过用户输入的标志来判断
   3. 源程序

#include <iostream>

using namespace std;

const char \*suit[4] = { "Spades", "Hearts", "Clubs", "Diamonds" };

const char \*face[13] = { "Ace", "Deuce", "Three", "Four", "Five", "Six",

"Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King" };

int deck[52];

int index = 0, i = 0;

void Swap(int &x, int &y);

int Func(void(\*p)(int times), int times);

void Shuffle(int times);

void Deal(int times);

int main() {

bool quit = false;

int times = 10;

char choice;

for (i = 0; i < 52; i++) {

deck[i] = i;

}

while (!quit) {

cout << "q: quit; 1: shuffle; 2: deal;" << endl;

cin >> choice;

switch (choice) {

case 'q':

quit = true;

break;

case '1':

cout << "How many times?" << endl;

cin >> times;

Func(Shuffle, times);

break;

case '2':

cout << "How many cards?" << endl;

cin >> times;

Func(Deal, times);

break;

default:

cout << "Please enter a valid value" << endl;

}

}

return 0;

}

void Swap(int &x, int &y) {

int temp = x;

x = y;

y = temp;

}

int Func(void(\*p)(int times), int times) {

p(times);

return 0;

}

void Shuffle(int times) {

for (i = 0; i < times; i++) {

Swap(deck[index + rand() % (52 - index)], deck[index + rand() % (52 - index)]);

}

}

void Deal(int times) {

int old\_index = index;

for (i = index; i < old\_index + times; i++) {

cout << face[deck[i] % 13] << " " << suit[deck[i] / 13] << " ";

if (index == 51) {

index = 0;

cout << "\nAll cards are dealt. Deal from beginning again." << endl;

break;

}

++index;

}

cout << endl;

}

1. 测试数据、运行结果

q: quit; 1: shuffle; 2: deal;

1

How many times?

50

q: quit; 1: shuffle; 2: deal;

2

How many cards?

10

Ace Spades Ten Diamonds Seven Hearts Eight Diamonds Six Hearts Four Diamonds Deuce Spades Three Diamonds Ten Hearts Three Hearts

q: quit; 1: shuffle; 2: deal;

1

How many times?

100

q: quit; 1: shuffle; 2: deal;

2

How many cards?

20

Seven Diamonds Seven Clubs Four Spades King Hearts Nine Clubs Queen Hearts Nine Spades Ace Diamonds Five Hearts Five Diamonds Jack Clubs Three Clubs Three Spades Ace Clubs Deuce Clubs Nine Diamonds Ten Spades Five Spades King Clubs Four Hearts

q: quit; 1: shuffle; 2: deal;

1

How many times?

1000

q: quit; 1: shuffle; 2: deal;

2

How many cards?

30

Seven Spades Ten Clubs Jack Diamonds Jack Spades Six Diamonds King Spades Eight Hearts Nine Hearts King Diamonds Six Spades Eight Spades Queen Spades Queen Clubs Four Clubs Ace Hearts Six Clubs Deuce Diamonds Five Clubs Deuce Hearts Queen Diamonds Jack Hearts Eight Clubs

All cards are dealt. Deal from beginning again.

q: quit; 1: shuffle; 2: deal;

2

How many cards?

40

Ace Spades Ten Diamonds Seven Hearts Eight Diamonds Six Hearts Four Diamonds Deuce Spades Three Diamonds Ten Hearts Three Hearts Seven Diamonds Seven Clubs Four Spades King Hearts Nine Clubs Queen Hearts Nine Spades Ace Diamonds Five Hearts Five Diamonds Jack Clubs Three Clubs Three Spades Ace Clubs Deuce Clubs Nine Diamonds Ten Spades Five Spades King Clubs Four Hearts Seven Spades Ten Clubs Jack Diamonds Jack Spades Six Diamonds King Spades Eight Hearts Nine Hearts King Diamonds Six Spades

q: quit; 1: shuffle; 2: deal;

q

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);

1. 分析

最初的程序使用的输出的方法是这样的：

cout << hex << modulo;

由于cout是一个流，所以hex这个参数会影响之后所有的内容，所以括号内的base本来为十进制的16也被转换为了16进制的10，而且这种方法也不能产生大写的F。

1. 总结

cout接受的是一个流, 所以前面的内容会对后面的内容产生影响，这时候我们就可以利用printf函数来避免这种影响 。

**三、实验小结**

1. 完成情况

将cout的输出用printf的输出方法替换之后得到正确结果。

1. 重点

递归函数函数的定义和调用。

1. 难点

如何将进制转换处理后的结果正确地输出并得到想要的格式的结果。

1. 解决方法

将cout << hex << modulo; 语句替换为printf("%X", modulo);。

1. 有待改进之处

printf用来输出并不是一个最佳的选择，很可能利用cout也能得到想要的结果

1. 收获、体会

合理地利用递归函数可以有效地简化问题，而且递归函数中的递归语句的位置要特别注意。C++中有许多不同的输出方法,比如cout 和printf,它们有各自的特点，我们可以利用这些特点来输出我们想要的格式。