电子科技大学信息与软件工程学院

**实 验 报 告**

学 号

姓 名

（实验） 课程名称 C语言程序设计

理论教师 张学

实验教师 张学

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名： 学号： 指导教师：张学**

**实验地点：科研楼A504 实验时间：16.11.24**

1. **实验名称：**C语言的程序结构和指针的用法
2. **实验学时：3学时**
3. **实验目的：**

1. 掌握C语言的程序结构。

2. 掌握C语言指针的用法。

3. 掌握C语言指针和数组的关系。

4. 掌握C语言字符串的用法。

1. **实验原理：**

1.局部变量

（1）在函数体内声明的变量称为该函数的局部变量。自动存储期限。局部变量的存储单位是在包含该变量的函数被调用时“自动”分配的，函数返回时收回分配；块作用域。局部变量的作用域是从变量声明的点开始一直到所在函数体的末尾。

（2）静态局部变量

在局部变量声明中放置单词 static可以使变量具有静态存储期限。

因为具有静态存储期限的变量拥有永久的存储单位，所以在整个程序执行期间都会保留变量的值。

（3）形式参数

形式参数拥有和局部变量一样的性质：自动存储期限和块作用域。在每次函数调用时，对形式参数自动进行初始化。

2. 外部变量

“参数传递”是向函数传递信息的一种方法。函数还可以通过外部变量进行通信——外部变量是声明在任何函数体外的。外部变量有时也被称为全局变量。

（1）外部变量的性质：静态存储期限；文件作用域：变量的可见范围从变量被声明的点开始一直到所在文件的末尾。

2.构建程序的方法。其中一种常用的顺序是：

#include 指令

#define 指令

定义类型

声明外部变量

声明除 main之外的函数原型

定义其它函数

定义 main函数

在函数定义之前增加一个注释框是一种很好的编程习惯。

3.指针变量：

用来存储内存地址。当采用指针变量p来存储变量i的地址时，我们通常说“p指向i”。

为修改传入函数的实际参数，可通过向函数传递指针（而不是变量值）来实现对实际参数的修改。可以使用const关键字确保函数不会修改指针参数所指向的对象。

指针可以采用关系运算符 (<, <=, >, >=) 和判等运算符 (== and !=)进行比较：只有在两个指针指向同一数组（中的元素）时，用关系运算符进行指针比较才有意义。指针比较的结果依赖于指针指向的数组元素在数组中的相对位置。

4.字符串字面量：用一对双引号括起来的字符序列。当编译器遇到一个长度为n的字符串字面量时，给该字符串分配n+1个字节的内存空间。该内存空间将存放字符串中的字符，外加一个额外的空字符，用于标志字符串的结束。

4.字符串变量

（1）任何一维字符数组均可以用于存储字符串，字符串必须以空字符结尾。额外增加的1用于给字符串结束符留出空间。

（2）写字符串：Printf函数可以用 %s 转换说明符来写一个字符串，直至遇到空字符.printf不是唯一可用于写字符串的函数。puts函数也可用于写字符串:puts(str);

（3）用scanf 和 gets读字符串：Scanf函数用%s 转换说明符读字符到一个字符数组:scanf("%s", str); str 在这里是一个指针，因此不必在str前面放置 & 运算符。调用 scanf时，该函数跳过空白，然后读入字符并存入str指向的空间，直至遇到一个空白字符。scanf 函数会存放一个空字符在字符串的后面。

gets函数的特点:读取输入不会跳过开始的空白；直到找到新行符才停止读入；不存储新行符，而用空字符代替。

5.函数用于完成数组的操作

（1）strcpy 函数原型:char \*strcpy(char \*s1, const char \*s2);

strcpy 拷贝字符串 s2 到字符串s1.

（2）strncpy(str1, str2, sizeof(str1) - 1);

str1[sizeof(str1)-1] = '\0';

第二条预计保证了str1总是以空字符结尾的。

（3）Strlen函数原型:

size\_t strlen(const char \*s);

strlen 返回字符串s的长度，不包括空字符.

（4）strcat 函数原型:

char \*strcat(char \*s1, const char \*s2);

strcat 追加字符串s2的内容到字符串s1的末尾.

（5）Strncat是strcat的安全但是较慢的版本.

strncat(str1, str2, sizeof(str1) - strlen(str1) - 1);

strncat 会以一个空字符结束str1 .

（6）strcmp 函数原型:

int strcmp(const char \*s1, const char \*s2);

strcmp 比较字符串 s1 和 s2, 根据s1是小于、等于或者大于s2来返回一个小于、等于或者大于0的值.

6.7. 字符串数组

存储字符串数组有多种方法.

一种方法是采用二维字符数组，每行一个字符串；

另一种是采用指针数组的方式

1. **实验内容：**

项目1：第10章编程题1

项目2：第11章编程题4

项目3：第12章编程题1a

项目4：第12章编程题1b

项目5：第12章编程题3

项目6：第13章编程题1

1. **实验器材（设备、元器件）：**

个人电脑一台

1. **实验步骤：**

1、明确项目需求

2、编写代码

3、编译代码

4、测试程序

5、根据测试结果对程序进行调试改进

1. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**

项目1

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#define STACK\_SIZE 100

int contents[STACK\_SIZE];

int top = 0;

void make\_empty(void);

bool is\_empty(void);

bool is\_full(void);

void push(int i);

int pop(void);

void stack\_overflow(void);

void stack\_underflow(void);

void make\_empty(void){

top = 0;

}

bool is\_empty(void){

return top == 0;

}

bool is\_full(void){

return top == STACK\_SIZE;

}

void push(int i){

if (is\_full())

stack\_overflow();

else

contents[top++] = i;

stack\_overflow();

else

contents[top++] = i;

}

int pop(void){

if (is\_empty()){

stack\_underflow();

return 0;

} else

return contents[--top];

}

void stack\_overflow(void){

printf("Stack overflow\n");

exit(EXIT\_SUCCESS);

}

void stack\_underflow(void){

printf("Stack underflow\n");

exit(EXIT\_SUCCESS);

}

int main() {

int ch;

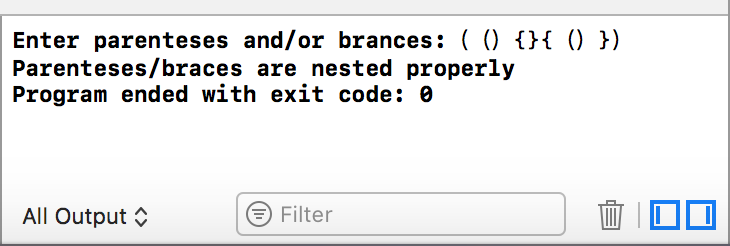
char c;

make\_empty();

printf("Enter parenteses and/or brances:");

while ((ch = getchar()) != '\n'){

测试结果：



实验结论：测试正确，程序满足项目需求。

if (ch == '(' || ch == '{' ){

push(ch);

}

else if (ch == ')' || ch == '}' ){

c = pop();

if ((c == '(' && ch == ')')||(c == '{' && ch == '}')){

continue;

}

else{

printf("Parenteses/braces are nested improperly\n");

exit(EXIT\_SUCCESS);

}

}

}

if (is\_empty()){

printf("Parenteses/braces are nested properly\n");

}

else {

printf("Parenteses/braces are nested improperly\n");

}

return 0;

}

项目2

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#define NUM\_RANKS 13

#define NUM\_SUITS 4

#define NUM\_CARDS 5

void read\_cards(int num\_in\_rank[], int num\_in\_suit[]);

void analyse\_hand(bool\*, bool\*, bool\*, bool\*, int\*, int[], int[]);

void print\_result(bool\*, bool\*, bool\*, bool\*, int\*);

int main(){

int num\_in\_rank[NUM\_RANKS];

int num\_in\_suit[NUM\_SUITS];

bool straight, flush, four, three;

int pairs;

for (; ;){

read\_cards(num\_in\_rank, num\_in\_suit);

analyse\_hand(&straight, &flush, &four, &three, &pairs, num\_in\_rank, num\_in\_suit);

print\_result(&straight, &flush, &four, &three, &pairs);

}

}

int pairs;

for (; ;){

read\_cards(num\_in\_rank, num\_in\_suit);

analyse\_hand(&straight, &flush, &four, &three, &pairs, num\_in\_rank, num\_in\_suit);

print\_result(&straight, &flush, &four, &three, &pairs);

}

}

void read\_cards(int num\_in\_rank[], int num\_in\_suit[])

{

bool card\_exists[NUM\_RANKS][NUM\_SUITS];

char ch, rank\_ch, suit\_ch;

int rank, suit;

bool bad\_card;

int cards\_read = 0;

for (rank = 0; rank < NUM\_RANKS; rank++) {

num\_in\_rank[rank] = 0;

for (suit = 0; suit < NUM\_SUITS; suit++)

card\_exists[rank][suit] = false;

}

for (suit = 0; suit < NUM\_SUITS; suit++)

num\_in\_suit[suit] = 0;

while (cards\_read < NUM\_CARDS) {

bad\_card = false;

printf("Enter a card: ");

rank\_ch = getchar();

switch (rank\_ch) {

case '0': exit(EXIT\_SUCCESS);

case '2': rank = 0; break;

case '3': rank = 1; break;

case '4': rank = 2; break;

case '5': rank = 3; break;

case '6': rank = 4; break;

case '7': rank = 5; break;

case '8': rank = 6; break;

case '9': rank = 7; break;

case 't': case 'T': rank = 8; break;

case 'j': case 'J': rank = 9; break;

case 'q': case 'Q': rank = 10; break;

case 'k': case 'K': rank = 11; break;

case 'a': case 'A': rank = 12; break;

default: bad\_card = true;

}

suit\_ch = getchar();

switch (suit\_ch) {

case 'c': case 'C': suit = 0; break;

case 'd': case 'D': suit = 1; break;

case 'h': case 'H': suit = 2; break;

case 's': case 'S': suit = 3; break;

default: bad\_card = true;

}

while ((ch = getchar()) != '\n')

if (ch != ' ') bad\_card = true;

if (bad\_card)

printf("Bad card; ignored.\n");

else if (card\_exists[rank][suit])

printf("Duplicate card; ignored.\n");

else {

num\_in\_rank[rank]++;

num\_in\_suit[suit]++;

card\_exists[rank][suit] = true;

cards\_read++;

}

}

}

void analyse\_hand(bool \*pstraight, bool \*pflush, bool \*pfour, bool \*pthree, int \*ppairs,int num\_in\_rank[], int num\_in\_suit[])

{

int num\_consec = 0;

int rank, suit;

\*pstraight = false;

\*pflush = false;

\*pfour = false;

\*pthree = false;

\*ppairs = 0;

/\* check for flush \*/

for (suit = 0; suit < NUM\_SUITS; suit++)

if (num\_in\_suit[suit] == NUM\_CARDS)

\*pflush = true;

/\* check for straight \*/

rank = 0;

while (num\_in\_rank[rank] == 0) rank++;

for (; rank < NUM\_RANKS && num\_in\_rank[rank] > 0; rank++)

num\_consec++;

if (num\_consec == NUM\_CARDS) {

\*pstraight = true;

return;

}

/\* check for 4-of-a-kind, 3-of-a-kind, and pairs \*/

for (rank = 0; rank < NUM\_RANKS; rank++) {

if (num\_in\_rank[rank] == 4) \*pfour = true;

if (num\_in\_rank[rank] == 3) \*pthree = true;

if (num\_in\_rank[rank] == 2) (\*ppairs)++;

}

}

void print\_result(bool \*pstraight, bool \*pflush, bool \*pfour, bool \*pthree, int \*ppairs)

{

if (\*pstraight && \*pflush) printf("Straight flush");

else if (\*pfour) printf("Four of a kind");

else if (\*pthree && \*ppairs == 1) printf("Full house");

else if (\*pflush) printf("Flush");

else if (\*pstraight) printf("Straight");

else if (\*pthree) printf("Three of a kind");

else if (\*ppairs == 2) printf("Two pairs");

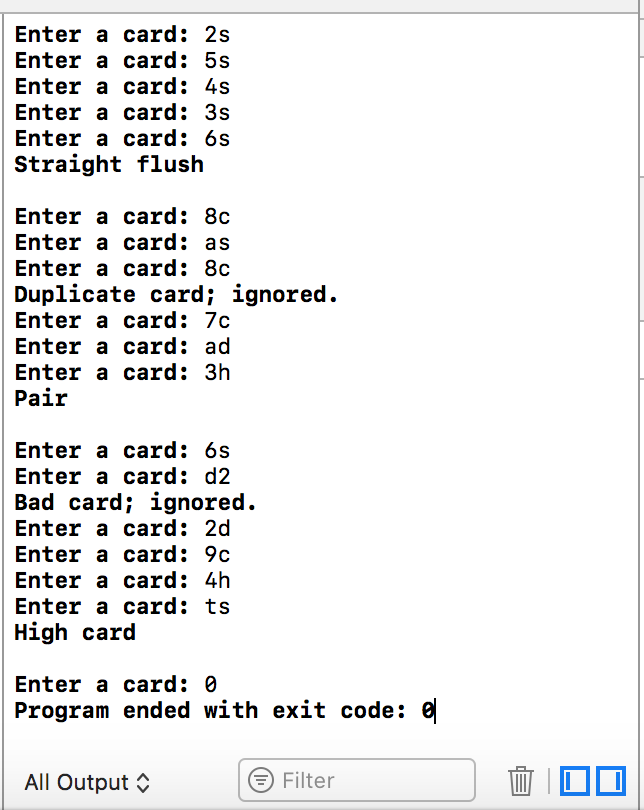
else if (\*ppairs == 1) printf("Pair");

else printf("High card");

printf("\n\n");

}

测试结果：



实验结论：测试正确，程序满足项目需求。

else if (\*pthree) printf("Three of a kind");

else if (\*ppairs == 2) printf("Two pairs");

else if (\*ppairs == 1) printf("Pair");

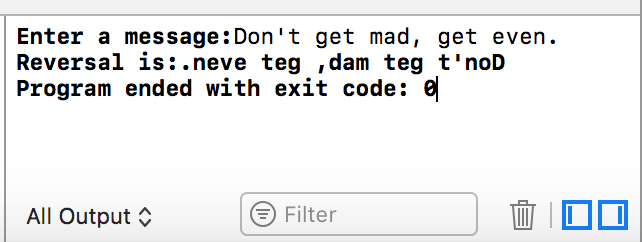
else printf("High card");

printf("\n\n");

}

项目3

测试结果：



实验结论：测试正确，程序满足项目需求。

#include <stdio.h>

int main() {

char c[99] = {'\0'};

char \*p;

int ch;

p = &c[0];

printf("Enter a message:");

while ((ch = getchar()) != '\n'){

\*p = ch;

p++;

}

printf ("Reversal is:");

for ( ; p >= &c[0]; p--){

putchar(\*p);

}

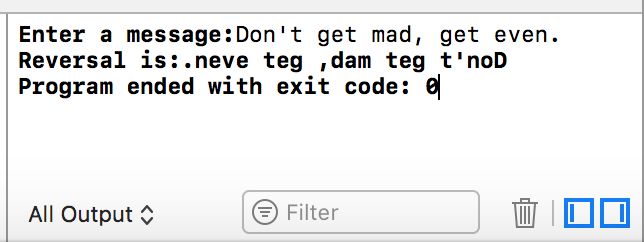
printf ("\n");

return 0;

}

项目4

测试结果：



实验结论：测试正确，程序满足项目需求。

#include <stdio.h>

int main() {

char c[99] = {'\0'};

int ch, i = 0;

printf("Enter a message:");

while ((ch = getchar()) != '\n'){

\*(c + i++) = ch;

}

printf ("Reversal is:");

for (;i--;){

putchar(\*(c+i));

}

printf ("\n");

return 0;

}

项目5

#include <stdio.h>

#include <string.h>

#define Max\_len 20

int main() {

unsigned long len;

char word[Max\_len];

char smallest\_word[Max\_len], largest\_word[Max\_len];

printf ("Enter word:");

scanf("%s", word);

len = strlen(word);

strcpy( smallest\_word, word);

strcpy( largest\_word, word);

while (len != 4){

printf ("Enter word:");

scanf("%s", word);

len = strlen(word);

if(strcmp(word, smallest\_word) < 0){

strcpy(smallest\_word, word);

}

if(strcmp(word, largest\_word) > 0){

strcpy(largest\_word, word);

}

if (len == 4){

break;

printf ("Enter word:");

scanf("%s", word);

len = strlen(word);

if(strcmp(word, smallest\_word) < 0){

strcpy(smallest\_word, word);

}

if(strcmp(word, largest\_word) > 0){

strcpy(largest\_word, word);

}

if (len == 4){

break;

}

}

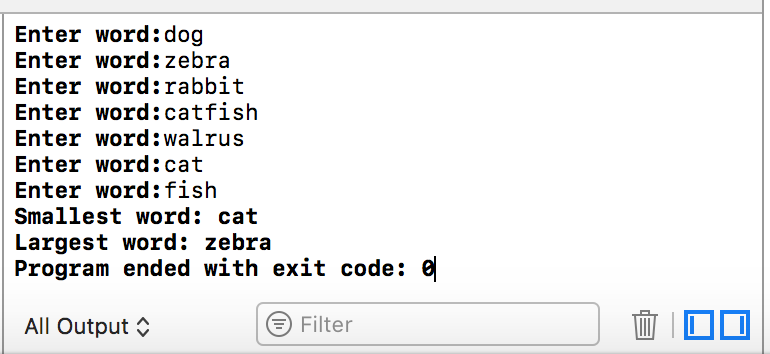
printf("Smallest word: %s\n", smallest\_word);

printf("Largest word: %s\n", largest\_word);

return 0;

}

测试结果：



实验结论：测试正确，程序满足项目需求。

1. **总结及心得体会：**

总结：项目1是通过定义一个栈来实现字符的存储和输出。编写了很多函数来实现栈的功能：清零，判断栈是否为空，是否为满，数据入栈，数据弹出栈。主函数采用循环和if来输入括号和判断是否匹配

项目2是将一个引用外部变量的函数改编成使用指针传参的函数。需要注意的就是数组名可以作为指针直接传递数组。需要改变值的函数必须用指针传参，否则值传递只能改变副本。

项目3是将原函数改为使用指针引用数组。定义指针后首先应该给指针赋值，然后采用指针的自增操作进行数组元素的移位。

项目4则在项目3运用指针的基础上使用数组名作为指针。数组名是数组中第一个元素的地址，采用加一个偏移变量i使其移位。

项目5考察了对字符串的运用。使用一个字符数组来存储每次输入的字符串，将其与最大值最小值字符串进行比较并找出最大最小的字符串，再判断其长度是否为4.

心得体会：此次实验主要是考察指针，数组和字符串的运用，涉及到c语言的比较深的精髓的地方。刚刚开始做实验的时候，直接开始做，没有头绪，比较混乱，后来仔细看了书和笔记复习了再看就柳暗花明，豁然开朗。所以知识重在理解，理解掌握了才能够熟练的运用。

1. **对本实验过程及方法、手段的改进建议：无**

**报告评分：**

**指导教师签字：**