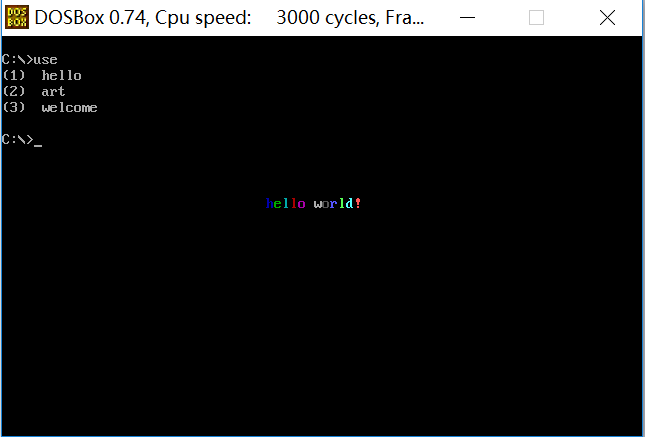
**c语言综合研究十三报告**

1. **研究过程展示**

**1>.程序1部分：**

原本的程序运行结果为：



程序use.c部分：

#include "menu.h"

void putstr(char\*);

char text1[20] = "(1) hello";

char text2[20] = "(2) art";

char text3[20] = "(3) welcome";

void f1();

void f2();

void f3();

ItemType item[3]={

text1,'1',f1,

text2,'2',f2,

text3,'3',f3

};

main()

{

menu(item,3);

}

void f1()

{

putstr("hello world!");

}

void f2()

{

putstr("the art of c");

}

void f3()

{

putstr("welcome to c");

}

void putstr(char\* str)

{

int far\* scr =(int far\*)(0xb8000000+160\*10+33\*2);

int n;

for(n = 0;str[n];n++)

{

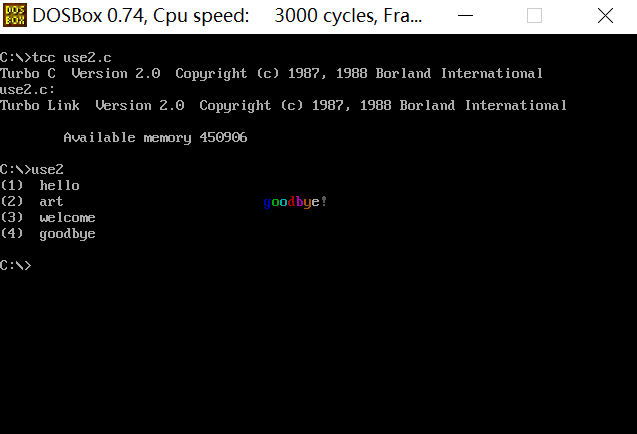
scr[n] = str[n]+(n+1)\*0x100;

}

}

**2>.程序2部分：**

添加第四个字符串“goodbye”，运行结果：



程序use2.c部分：

#include "menu2.h"

void putstr(char\*);

char text1[20] = "(1) hello";

char text2[20] = "(2) art";

char text3[20] = "(3) welcome";

char text4[20] = "(4) goodbye";

void f1();

void f2();

void f3();

void f4();

ItemType item[4]={

text1,'1',f1,

text2,'2',f2,

text3,'3',f3,

text4,'4',f4

};

main()

{

menu(item,4);

}

void f1()

{

putstr("hello world!");

}

void f2()

{

putstr("the art of c");

}

void f3()

{

putstr("welcome to c");

}

void f4()

{

putstr("goodbye!");

}

void putstr(char\* str)

{

int far\* scr =(int far\*)(0xb8000000+160\*10+33\*2);

int n;

for(n = 0;str[n];n++)

{

scr[n] = str[n]+(n+1)\*0x100;

}

}

分析部分：

但是在自我添加“goodbye”这一“个性”的时候，我们可以发现，所有的修改都是发生在use.c文件中进行的。而menu.h文件中的内容没有丝毫修改。感觉是把程序的共性封装在了menu.h文件中，个性在use.c中，两者区别开来。

如果是仿照这样数据处理的话，感觉就像是在研究12中的b.c程序基础上对个性和共性进行划分就行了。

如下：

13demo1.h:

char a[20];

char b[20];

char ch;

int n;

void gongxing();

void gongxing()

{

gets(a);

printf("%c\n",ch = getch());

gets(b);

for(n = 0;code[n] && code[n] != ch;n++ );

if(!code[n] )

{

printf("error!");

return;

}

printf("--------------------\n");

}

13demo1.c:

#include "13demo1.h"

char\* code = "+-\*/";

int add(int a,int b)

{

return a+b;

}

int sub(int a,int b)

{

return a-b;

}

int mul(int a,int b)

{

return a\*b;

}

int div(int a,int b)

{

return a/b;

}

int (\*func[4])(int,int) = {add,sub,mul,div};

main()

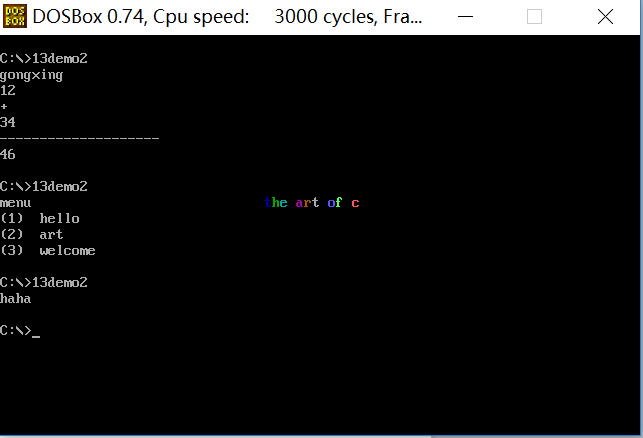
{

gongxing();

printf("%d",func[n](atoi(a),atoi(b)));

}

将计算和显示字符合体编一下：



程序：

13demo2.h:

char a[20];

char b[20];

char ch;

char choice[20];

char\* code = "+-\*/";

int n;

typedef struct{

char\* text;

char key;

void (\*func)();

}ItemType;

void menu(ItemType\* ,int);

void gongxing();

void gongxing()

{

gets(a);

printf("%c\n",ch = getch());

gets(b);

for(n = 0;code[n] && code[n] != ch;n++ );

if(!code[n] )

{

printf("error!");

return;

}

printf("--------------------\n");

}

void menu(ItemType\* item,int n)

{

int a,b;

char ch;

for(a = 0;a < n;a++)

{

b =strlen(item[a].text);

if(b > 20 || b <0)

{

printf("error");

return;

}

printf("%s\n",item[a].text);

}

do{

ch = getch();

for(a = 0;item[a].key != ch && a < n ;a++);

}while(a == n);

item[a].func();

}

13demo2.c:

#include "13demo2.h"

char text1[20] = "(1) hello";

char text2[20] = "(2) art";

char text3[20] = "(3) welcome";

void f1();

void f2();

void f3();

void putstr(char\*);

ItemType item[3]={

text1,'1',f1,

text2,'2',f2,

text3,'3',f3

};

int add(int a,int b)

{

return a+b;

}

int sub(int a,int b)

{

return a-b;

}

int mul(int a,int b)

{

return a\*b;

}

int div(int a,int b)

{

return a/b;

}

int (\*func[4])(int,int) = {add,sub,mul,div};

void f1()

{

putstr("hello world!");

}

void f2()

{

putstr("the art of c");

}

void f3()

{

putstr("welcome to c");

}

void putstr(char\* str)

{

int far\* scr =(int far\*)(0xb8000000+160\*10+33\*2);

int n;

for(n = 0;str[n];n++)

{

scr[n] = str[n]+(n+1)\*0x100;

}

}

main()

{

gets(choice);

if( 0 == strcmp(choice,"menu") )

{

menu(item,3);

}

if( 0 == strcmp(choice,"gongxing") )

{

gongxing();

printf("%d\n",func[n](atoi(a),atoi(b)));

}

}

1. **已思考研究并已解决问题汇总**
2. **已思考研究并未解决问题汇总**
3. **研究感想（心得体会）**

其实\*.c文件和其include的\*.h文件从功能上看视作同一个文件，是一个整体，但是对程序中的各个部分进行共性和个性的区分，然后再进行放置。可以实现有序性编程。看过那个“尼古拉斯.赵四”跳舞的那个gif图，跳着跳着自己的腿把自己绊倒了。现实中也一样，经常搞混我们的，往往就是我们自己。