|  |
| --- |
| 程序设计训练  日记与报告 |
|  |

专业： 电子信息工程

班级： 信息182班

学号： 201811050897

姓名： 王 澳 澳

教师： 李京胜、 禚钊

信息与控制工程学院

2020年12月

**程序设计训练任务书**

**一、实习的任务、性质与目的**

程序设计训练是在学生学习了程序设计基础（C）、数据结构、面向对象程序设计（C++）、数据库等软件开发相关课程后，针对信息类专业需要较强的程序设计能力而进行的一次程序设计方面的集中综合训练。其目的是使学生理解和掌握软件设计的过程和步骤，学会基本程序调试技巧和方法，掌握现代开发工具、开发流程与规范，具备共同探讨技术性问题的能力，加强学生自主学习、收集资料和动手编程的能力能够运用系统性的、规范化的、可定量的过程化方法去开发和维护通信领域的软件。

1. **实习内容**
2. **基础部分**
3. **函数问题（数的合并）**

编写一个能将两个数合并成一个数的函数fun，具体合并方式是：将两个两位数的正整数a和b合并形成一个整数放在c中，并将a的十位和个位依次存放在c数的十位和千位上，b的十位和个位依次存放在c数的个位和百位上，基本要求：在主函数中调用fun函数，fun函数要写出以下三种：（1）通过返回值将新数c传给主函数；（2）通过引用参数将新数传给主函数；（3）通过指针型参数将新数传给主函数。

代码：

1）通过返回值将新数c传给主函数；

*#include<stdio.h>*

int main()

{ int fun(int a,int b);

int a,b,c;

printf("Please enter a:\n");

scanf("%d",&a); //接收正整数a

printf("Please enter b:\n"); scanf("%d",&b); //接收正整数b

c=fun(a,b);

printf("The number c is %d",c); //输出正整数c

return 0; }

int fun(int a,int b)

{ int c1,c2,c3,c4; //c1、c2、c3、c4分别对应c的千位、百位、十位、个位

c1=a%10;

c2=b/10;

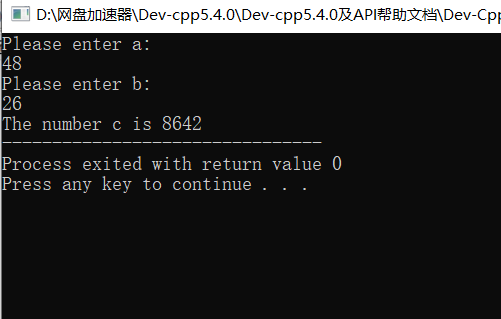
c3=a/10;

c4=b%10;

return (c1\*1000+c2\*100+c3\*10+c4);

}

运行截图：



2）通过引用参数将新数传给主函数；

#include<stdio.h>

int fun(int a, int b)

{

int a1,a10,b1,b10,c;

a10=a/10;

a1=a%10;

b10=b/10;

b1=b%10;

c=a1\*1000+b1\*100+a10\*10+b10\*1;

return c;

}

int main()

{

int m,n;

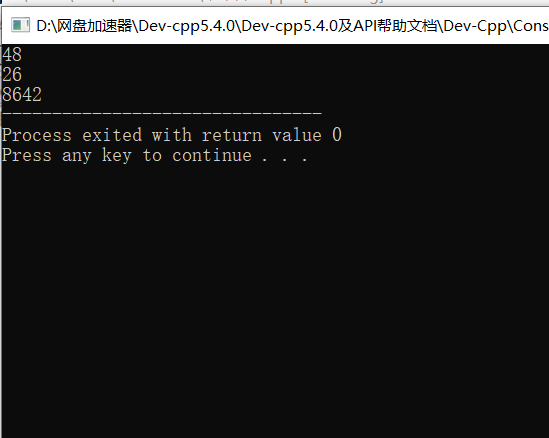
scanf("%d%d",&m,&n);

printf("%d",fun(m,n));

return 0;

}

运行截图：



3）通过指针型参数将新数传给主函数；

#include "stdio.h"

int fun(int \*x,int \*y)

{

int t,m,n,o;

t=\*x/10;

m=\*x%10;

n=\*y/10;

o=\*y%10;

return (m\*1000+o\*100+t\*10+n);

}

int main()

{

int a,b,y;

printf("请输入a和b：");

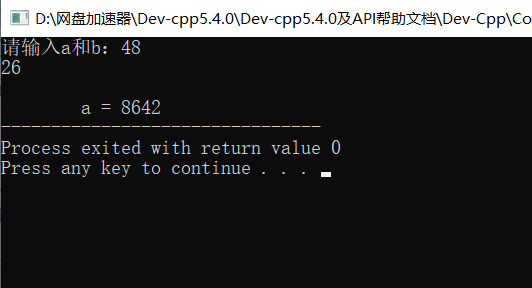
scanf("%d%d",&a,&b);

y=fun(&a,&b);

printf("\n\ta = %d",y);

}

运行截图：



总结：三个方法各有优缺点。通过返回值将新数c传给主函数方法代码过多；通过引用参数将新数传给主函数方法是我本人喜欢使用的一种，方法简单易懂；通过指针型参数将新数传给主函数方法要明白指针的输入存储，也通俗易懂。

**2、线性表问题（猴子选大王）**

一堆猴子都有编号，编号是1，2，3 ...m，这群猴子（m个）按照1-m的顺序围坐一圈，从第1开始数，每数到第N个，该猴子就要离开此圈，这样依次下来，直到圈中只剩下最后一只猴子，则该猴子为大王。

基本要求：

（1） 输入数据：输入m,n m,n 为整数，n<m

（2）中文提示按照m个猴子，数n个数的方法，输出为大王的猴子是几号 ，建立一个函数来实现此功能

（3）分别用数组和链表来实现，并比较分析不同存储结构下的时间复杂度和空间复杂度。

设m=25,n=4

问题分析：

程序代码实利用循环，当只剩一只猴子时循环停止，淘汰掉的猴子值为0，每数到n就要跳过值为0的猴子，即如果值为0则继续往后输一个猴子，让当前计数为0并将这个猴子的值改为0，同事淘汰掉的猴子数加1，最后遍历所有的猴子找到值不为0的即为猴子大王。

代码：

数组

#include <iostream>

#include <vector>

using namespace std;

const int N = 25;

//打印N只猴子的状态（“1”表示在圈内，“0”表示不在圈内）

void printVec(const vector<int> &v)

{

for (vector<int>::const\_iterator it = v.begin();it != v.end();++it)

{

cout<<\*it<<' ';

}

cout<<endl;

}

int monkeyKing(vector<int> v)

{

//n记录还有多少只猴子在圈内

int n = N;

int k = 0;

int i;

while (n > 1)

{

for (i=0;i<N;++i)

{

k += v[i];

if (k == 4)

{

v[i] = 0;

printVec(v);

k = 0;

--n;

} } }

//最后一只猴子下标为1

for (int i=0;i<N;++i)

{

if (1 == v[i])

return i+1;

}}

int main()

{

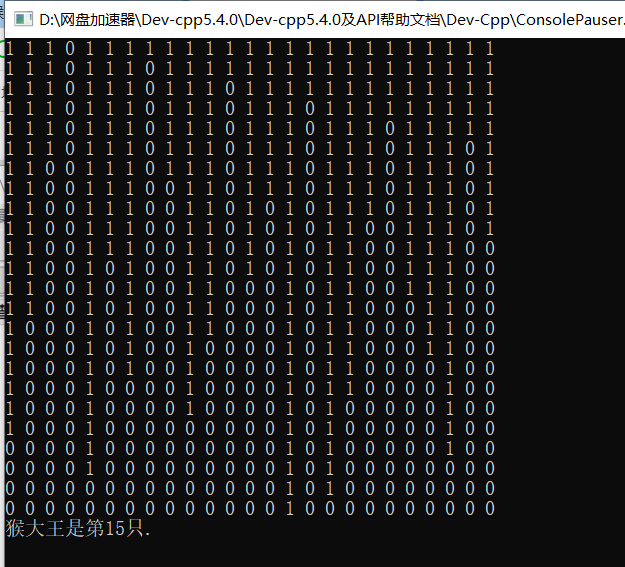
vector<int> v(N,1);

cout<<"猴大王是第"<<monkeyKing(v)<<"只."<<endl;

return 0;

}

运行截图：



链表

#include <iostream>

using namespace std;

typedef struct test

{

int id;

test \*next;

};

test \* newnode;

/\*删除后返回当前结点\*/

test\* delete\_node(test \*headnode, test \*deletenode)

{

if (headnode->next == NULL)

return NULL;

test \* f\_head = headnode;

test \* f\_head\_next = f\_head->next;

while (f\_head\_next != NULL)

{

if (f\_head\_next == deletenode)

{

if (f\_head\_next->next == NULL)

{

f\_head->next = NULL;

cout<<"Null ="<<headnode->id<<endl;

}

else

{

f\_head->next = f\_head\_next->next;

return f\_head\_next->next;

}

break;

}

f\_head = f\_head->next;

f\_head\_next = f\_head->next;

}

}

// 增加新结点

test \* addnode(test \* node, int idnum)

{

newnode = new test;

newnode->id = idnum;

newnode->next = NULL;

node->next = newnode;

return newnode;

}

//排序

void sort(test \*head)

{

if (head->next == NULL)

return;

int temp;

test \* headnode = head;

test \* second = headnode->next;

while (headnode != NULL)

{

second = headnode->next;

while (second != NULL)

{

if ((headnode->id )> (second->id))

{

temp = headnode->id;

headnode->id = second->id;

second->id = temp;

}

second = second->next;

}

headnode = headnode->next;

}

}

int main(void)

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*begin\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

test head;

int nodenum = 0 ;

test \*temp = &head;

test \*currentNode = &head; //此处变量相当于中间变量，为链表当前结点位置

test \*before\_node = head.next;

bool ishead = false;

cout<<"请输入链表结点值："<<endl;

while (nodenum ++ != 30)

{

if (!ishead)

{

head.id =nodenum;

ishead = true;

continue;

}

before\_node = currentNode; /\*保存指针的前一个位置\*/

currentNode = addnode(currentNode, nodenum);

}

sort(temp);

temp = &head;

int i = 1;

cout<<"链表的值为："<<endl;

while (temp!= NULL)

{

cout<<"temp.id "<<i<<" = "<< temp->id<<endl;

temp = temp->next;

i++;

}

/\*尾部指针指向头指针\*/

currentNode->next = &head;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

//before\_node = delete\_node(&head, before\_node);

//cout<<"new node = "<<before\_node->id<<endl;

/\*开始选大王\*/

temp = &head;

i = 0;

int node\_num = 25;

while (node\_num != 1)

{

i++;

if (4 == i)

{

cout<<"删除结点为："<<temp->id<<endl;

temp = delete\_node(&head, temp);

i = 0;

node\_num--;

}

else

temp = temp->next;

}

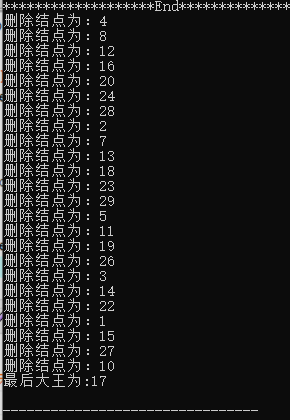
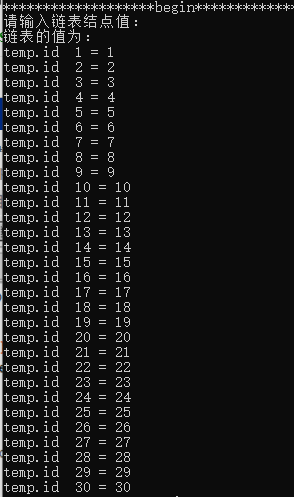
cout<<"最后大王为:"<<temp->id<<endl;

delete [] newnode; //释放

return 0;

}

运行截图：



总结：

数组中代表的第几个数实际上是下标再加1，通过结果很明显可以看到通过链表解决该问题最适合不过，时间和空间复杂度明显要比数组要小很多。

**3、排序问题（奖学金）**

先根据输入的3门课的成绩计算总分，然后按上述规则排序，最后按排名顺序输出前五名名学生的学号和总分。注意，在前5名同学中，每个人的奖学金都不相同，因此，你必须严格按上述规则排序。

问题分析：

先通过求和计算比较出总分高的前三名同学，再比较前三名中总分是否相同，若相同，就在进行语文分数比较，语文分数高者排名靠前，若相同，学号小者排名靠前。

代码：

#include <stdio.h>

#include <iostream>

#include <algorithm>

using namespace std;

struct Stu {

int id;

int chinese;

int math;

int english;

int sum;

}stu[305];

bool cmp(struct Stu a, struct Stu b) {

if(a.sum != b.sum) {

return a.sum > b.sum;

} else {

if(a.chinese != b.chinese) {

return a.chinese > b.chinese;

} else {

return a.id < b.id;

}

}

}

int main() {

int n, i;

scanf("%d", &n);

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

stu[i].id = i+1;

scanf("%d %d %d", &stu[i].chinese, &stu[i].math, &stu[i].english);

stu[i].sum = stu[i].chinese + stu[i].math + stu[i].english;

}

sort(stu, stu+n, cmp);

for(i = 0; i < 5; ++i) {

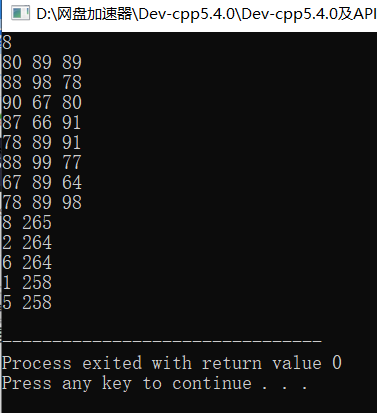
printf("%d %d\n", stu[i].id, stu[i].sum);

}

return 0;

}

运行截图：



**4、字符串问题**

地图上的每个点有三种状态未感染、被感染和被保护，分别用大写字母O,X,P表示。一段病毒周期之内一个感染区域会感染上、下、左、右四个方向上的区域，被保护的区域则不会被传染。输入一个地图和经历的病毒周期数，推算出经历了这些病毒周期后的新地图。

问题分析：

先建立一个二维数组，并且全部进行初始化为O，然后输入安全区以及感染区，利用两个for循环将数组所有数据判断感染区在哪，并且将该区域的上下左右进行修改为X，此为第一次疫情过后，第二次同理。在该循环外，再套用一个for循环，进行疫情次数的输入和疫情情况的输出。

代码：

#include <iostream>

using namespace std;

int main()

{

int M,N,i,j,c;

cout<<"请输入行数M: "<<endl;

cin>>M;

cout<<"请输入列数N: "<<endl;

cin>>N;

char a[100][100],b[100][100]={'A'};

cout<<"请输入矩阵："<<endl;

for(i=0;i<M;i++)

for(j=0;j<N;j++)

cin>>a[i][j];

cout<<"请输入病毒周期c:"<<endl;

cin>>c;

while(c--)

{

for(i=0;i<M;i++)

for(j=0;j<N;j++)

if(a[i][j]=='X')

b[i][j]=a[i][j];

for(i=0;i<M;i++)

for(j=0;j<N;j++)

if(a[i][j]==b[i][j])

{

if(a[i-1][j]!='P') a[i-1][j]='X';

if(a[i][j-1]!='P') a[i][j-1]='X';

if(a[i][j+1]!='P') a[i][j+1]='X';

if(a[i+1][j]!='P') a[i+1][j]='X';

}

}

for(i=0;i<M;i++)

{

cout<<endl;

for(j=0;j<N;j++)

cout<<a[i][j];

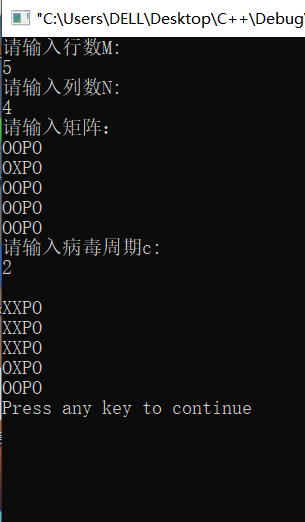
}

cout<<endl;

return 0;

}

运行截图：



总结：

矩阵的代码实现，以及安全区的判断和感染区的感染。通过每一次的解析思路和代码结果推测，一步一步实现成功。

**5、栈问题（多项式计算器）**

输入一个简单多项式，输出结果。多项式可能包含对整数的加减两种运算，中间没有空格。

输入

一行，一个多项式，保证输入合法

输出

一个结果

输入样例：

((1+2)\*3+4)+5+6\*7

输出样例：

60

问题分析：

输入一行表达式，若表达式有误，则输出表达式错误，否则计算出表达式的值并输出。运算符包括加减乘除，括号为小括号，但可以层层嵌套。输入表达式按回车结束。若计算结果为整数，则输出整数，若有小数，则输出浮点数。

代码：

#include <iostream>

#include <string>

#include <stack>

using namespace std;

double calculate(string s)

{

stack<double> num;

stack<char> op;

string temp = "";

for(int i = 0; i < s.length(); i++)

{

if(s[i] >= '0' && s[i] <= '9')

{

temp += s[i];

}

if((s[i] < '0' || s[i] > '9') && temp != "")

{

num.push(atoi(temp.c\_str()));

temp = "";

}

if(s[i] == '(' || s[i] == '[' || s[i] == '{')

{

op.push(s[i]);

}

if(s[i] == '\*' || s[i] == '/')

{

if(!op.empty())

{

char lastop = op.top();

if(lastop == '/')

{

double secondnum = num.top();

num.pop();

double firstnum = num.top();

num.pop();

num.push(firstnum\*1.0/secondnum);

op.pop();

}

}

op.push(s[i]);

}

if(s[i] == '+' || s[i] == '-')

{

if(i == 0 || s[i-1] == '(' || s[i-1] == '[' || s[i-1] == '{')

{

num.push(0);

}

if(!op.empty())

{

char lastop = op.top();

while(!op.empty() && (lastop == '\*' || lastop == '/'))

{

double secondnum = num.top();

num.pop();

double firstnum = num.top();

num.pop();

if(lastop == '\*')

num.push(firstnum\*secondnum);

else if(lastop == '/')

num.push(firstnum\*1.0/secondnum);

op.pop();

if(!op.empty())

lastop = op.top();

}

if(lastop == '-')

{

double secondnum = num.top();

num.pop();

double firstnum = num.top();

num.pop();

num.push(firstnum-secondnum);

op.pop();

}

}

op.push(s[i]);

}

if(s[i] == ')')

{

if(!op.empty())

{

char lastop = op.top();

while(!op.empty() && lastop != '(')

{

double secondnum = num.top();

num.pop();

double firstnum = num.top();

num.pop();

switch(lastop)

{

case '+':

num.push(firstnum+secondnum);

break;

case '-':

num.push(firstnum-secondnum);

break;

case '\*':

num.push(firstnum\*secondnum);

break;

case '/':

num.push(firstnum\*1.0/secondnum);

break;

}

op.pop();

if(!op.empty())

lastop = op.top();

}

if(lastop == '(')

op.pop();

}

}

if(s[i] == ']')

{

if(!op.empty())

{

char lastop = op.top();

while(!op.empty() && lastop != '[')

{

double secondnum = num.top();

num.pop();

double firstnum = num.top();

num.pop();

switch(lastop)

{

case '+':

num.push(firstnum+secondnum);

break;

case '-':

num.push(firstnum-secondnum);

break;

case '\*':

num.push(firstnum\*secondnum);

break;

case '/':

num.push(firstnum\*1.0/secondnum);

break;

}

op.pop();

if(!op.empty())

lastop = op.top();

}

if(lastop == '[')

op.pop();

}

}

if(s[i] == '}')

{

if(!op.empty())

{

char lastop = op.top();

while(!op.empty() && lastop != '{')

{

double secondnum = num.top();

num.pop();

double firstnum = num.top();

num.pop();

switch(lastop)

{

case '+':

num.push(firstnum+secondnum);

break;

case '-':

num.push(firstnum-secondnum);

break;

case '\*':

num.push(firstnum\*secondnum);

break;

case '/':

num.push(firstnum\*1.0/secondnum);

break;

}

op.pop();

if(!op.empty())

lastop = op.top();

}

if(lastop == '{')

op.pop();

}

}

}

if(temp != "")

num.push(atoi(temp.c\_str()));

while(!op.empty())

{

char lastop = op.top();

double secondnum = num.top();

num.pop();

double firstnum = num.top();

num.pop();

switch(lastop)

{

case '+':

num.push(firstnum+secondnum);

break;

case '-':

num.push(firstnum-secondnum);

break;

case '\*':

num.push(firstnum\*secondnum);

break;

case '/':

num.push(firstnum\*1.0/secondnum);

break;

}

op.pop();

}

if(num.empty())

return 0;

double value = num.top();

num.pop();

return value;

}

void test1()

{

string s;

while(cin >> s)

{

cout << calculate(s) << endl;

}

}

int main()

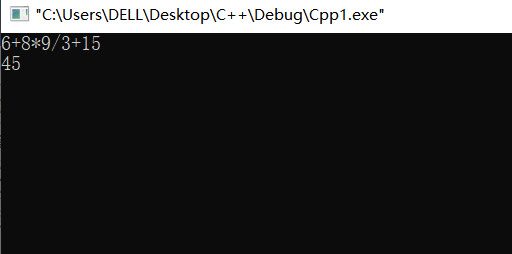
{

test1();

return 0;

}

运行截图：



**6、面向对象问题**

编写一个程序，有一个汽车类vehicle,它具有一个需传递参数的构造函数，类中的数据成员：车轮个数wheels和车重weight放在保护段中；小车类car是它的私有派生类，其中包含载人数passenger\_load；卡车类truck是vehicle的私有派生类，其中包含载人类passenger\_load和载重量payload。每个类都有相关数据的输出方法。

问题分析：

利用C++类和对象的知识，构造出vehicle类。新类可在基类所提供的的基础上包含新的成员，在新类中可隐藏基类的任何函数成员。

代码：

#include<iostream>

using namespace std;

class vehicle

{

protected:

int wheels;

float weight;

public:

vehicle(int w1, float w2)

{

wheels = w1;

weight = w2;

}

void printf()

{

cout << "wheels=" << wheels << "," << "weight=" << "weight" << endl;

}

};

class car :private vehicle

{

int passenger\_load;

public:

car(int w1,float w2,int p1):vehicle(w1,w2)

{

passenger\_load = p1;

}

void printf()

{

vehicle::printf();

cout << "passenger\_load=" << passenger\_load << endl;

}

};

class truck :private vehicle

{

int passenger\_load;

float payload;

public:

truck(int w1,float w2,int p1,float p2):vehicle(w1,w2)

{

passenger\_load = p1;

payload = p2;

}

void printf()

{

vehicle::printf();

cout << "passenger\_load=" << passenger\_load << "," << "payload=" << payload << endl;

}

};

void main()

{

vehicle v(1000, 842.2);

v.printf();

car c(134, 94.7, 84);

c.printf();

truck t(48,29.6,27,19.8);

t.printf();

}

运行截图：

