寻找自我的博客

编程珠玑第三章 数据决定程序结构

分类： 读书笔记 2012-08-13 17:11 53人阅读 评论(0) 收藏 举报

恰当的数据视图实际上决定了程序的结构。

书上介绍了一种编程方法： 格式信函编程。

它是把数据从控制中分离出来。用模板来实现数据的分离。

几条原则：

使用数组重新编写重复的代码

封装复杂结构

尽可能使用高级工具

从数据得出程序的结构

1 本书出版之时，美国的个人收入所得税分为5种不同的税率，其中最大的税率大约为40%.以前的情况则更为复杂，税率也更高。下面所示的程序文本采用25个if语句的合理方法来计算1978年的美国联邦所得税。税率序列为0.14, 0.15, 0.16, 0.17, 0.18.....。序列中此后的计算大于0.01.有何建议呢？

```
if income <= 2200
tax = 0;
else if income <= 2700
tax = 0.14 * (income - 2200);
else if income <= 3200
tax = 70 + 0.15 * (income - 2700);
else if income <= 3700
tax = 145 + 0.16 * (income -3200);
else if income <= 4200
tax =225 + 0.17 * (income - 3700);
.....
else
tax =53090 + 0.70 * (income - 102200);
```

采用二分搜索来定位分段函数。

```
#include <iostream>
using namespace std;

int basetax[100];
int lowerbound[100];
double taxrate[100];

int search(int lowerbound[],int income)
{
            int i=0;
            int j=99;
            int t=(i+j)/2;

            while(1)
```

```
            {
                        if(income - lowerbound[t] < 50 && income - lowerbound[t] >=
                                    return t;
                        else if(income - lowerbound[t] < 0) //在左侧寻找
                                    {
                                                j=t;
                                                t=(i+j)/2;

                                    }
                        else
                                    {
                                                i=t;
                                                t=(i+j)/2;
                                    }
            }
            return  -1;

}
int main()
{
            basetax[0]=0;
            lowerbound[0]=0;
            taxrate[0]=0;

            basetax[1]=0;
            lowerbound[1]=2200;
            taxrate[1]=0.14;

            for(int i=2;i<100;++i)
            {
                        basetax[i]=75*i-80;
                        lowerbound[i]=2200 + (i-1)*500;
                        taxrate[i]=(double)(14 + i-1)/100;

            }

            if(search(lowerbound,salary))
            {
                        int salary=2202;
                        int j=search(lowerbound,salary);

                        double tax= basetax[j] +   (double)taxrate[j]*(salary -lowe

                        cout<<tax<<endl;
            }

            return 0;
}
```

2.K阶常系数线性递归。

```
#include <iostream>
using namespace std;

int main()
{
            int t=0;
            int i,k;
    int  n=10;
            int c[10]={1,2,3,4,5,6,7,8,9,10};
            int a[10]={0};
```

```
            for(k=1;k<n;++k)
            {
            for(i=1;i<k;++i)
                        a[k] = a[k-i] * c [i];

                        a[k] +=c[k+1];
            }
            for(i=0;i<n;++i)
                        cout<<a[i]<<endl;

            return 0;
}
```

3.编写一个banner函数，该函数的输入为大写字母，输出为一个字符数组，该字符数组以图形化的方式来表示该字母。

遇到这种 输入数据很多，而且没有规律，就可以用 格式信函发生器用来用于解析格式信函模板。将数据从控制层分离的好处在于：避免每次针对不同的数据编写不同的代码；当需要改变一些公用文本的输出方式时，直接编辑模板即可，并不需要对数据进行修改。

对于26个字母，每个字母的外形并没有必然规律可循，最直接的方法是编写26个函数，针对特定的字母编写特定的打印程序，这是个体力活，代码数量将非常巨大。联想上面的格式信函编程，可以考虑为字母的外形设计一个定制模板，自己规定一套模板编写的格式，然后写一个解析程序，每次打印字母时，只需解析字母对应的模板即可，这样主要的工作量就花在每个字母模板的编写上，当然模板的编写是相当简单的，将字母图形转化为相应的模板格式即可。例如： 一个字母可以利用length = 12， width = 9的矩阵来表示

```
x  x  x  x  x  x  x  x  x
x  x  x  x  x  x  x  x  x
x  x  x  x  x  x  x  x  x
         x  x  x
         x  x  x
         x  x  x
         x  x  x
         x  x  x
         x  x  x
x  x  x  x  x  x  x  x  x
x  x  x  x  x  x  x  x  x
x  x  x  x  x  x  x  x  x
```

任何字母都可以在这张表示出来，每个点就像一个像素点。下面就对字母I和L进行模板编码，编码要求

（1）用尽可能简单的方式表示上面的图像；

（2）方便程序解析；

（3）必须适用于所有的情况

根据书上给出的编码结构，上图可表示为：

39x

63b3x3b

39x

编码规则：第一列表示要打印的行数,，后面的数字代表每行要打印的字符个数，个数后面紧跟要打印的字符，并用空格隔开。这里字母b表示空格。根据上述规则，字母L编码如下：

93x6b

39x

```
x   x   x
x   x   x
x   x   x
x   x   x
x   x   x
x   x   x
x   x   x
x   x   x
x   x   x
x   x   x   x   x   x   x   x   x
x   x   x   x   x   x   x   x   x
x   x   x   x   x   x   x   x   x
```

4. 日期处理

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int days[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
typedefstruct _date
{
int year, month, day;
}date;

//看是否是闰年
int leap(int year)
{
return (0 == year % 4 && year % 100) || year % 400 == 0;
}

//看是时间是否有效
int legal(date a)
{
if (a.month <0 || a.month > 12) return 0;
if (a.month == 2) return a.day > 0 && a.day <= (28 + leap(a.year));
return a.day >0 && a.day <= days[a.month -1];
}


//比较时间
int datecmp(date a, date b)
{
if (a.year != b.year) return a.year - b.year;
if (a.month != b.month) return a.month - b.month;
return a.day - b.day;
}


//计算周几?
int weekday(date a)
{
inttm = a.month >= 3 ? (a.month - 2) : (a.month + 10);
int ty = a.month >= 3 ? (a.year) : (a.year - 1);
return (ty + ty/4 - ty/100 + ty/400 + (int)(2.6 * tm - 0.2) + a.day) % 7;
}
```

```
//时间化为天数
int date2int(date a)
{
  int i ;
int ret = a.year * 365 + (a.year - 1) / 4 - (a.year -1) / 100 + (a.year - 1) / 400
  days[1] += leap(a.year);
for (i = 0; i < a.month - 1; ret += days[i++]);
  days[1] = 28;
 return ret + a.day;
}

//天数化为时间
date int2date(int a)
{
 date ret;
 int i = 0;
ret.year = a/146097*400;
for (a %= 146097; a >= 365+leap(ret.year);
 a -= 365 + leap(ret.year), ret.year++);
 days[i] += leap(ret.year);
for (ret.month = 1; a >= days[ret.month - 1]; a -= days[ret.month - 1], ret.month+
  days[1] = 28;
ret.day = a + 1;
 return ret;
}

//计算距离
int dist(date a, date b)
{
 int ad = date2int(a);
 int bd = date2int(b);
return ad - bd > ? ad - bd : bd - ad;
}


//生成日历
void cal(date a)
{
int i, w, j, k;
 a.day = 1;
if (a.month == 2) { k = days[a.month - 1] + leap(a.year);}
 else k = days[a.month - 1];

printf(" %2d月%4d年 \n", a.month, a.year);
 printf("日 一 二 三 四 五 六\n");
 w = weekday(a);
i = w % 7;while (i--) printf(" "); printf("%2d", 1);
 if (w % 7 == 6) printf("\n"); ++w;

for (i = 1; i <= k; ++i, w = (w + 1) % 7)
{
 if (w % 7 == 0) printf("%2d", i);
 else printf(" %2d", i);
 if (w % 7 == 6) printf("\n");
}
}
```

5.单词后缀处理。

```
#include<iostream>
```

```
#include<string>
#include<algorithm>
#include<hash_map>
#include<hash_set>
#include<iterator>

using namespace std;
using namespace stdext;

char *p[] ={"et-ic","al-is-tic","s-tic","p-tic","-lyt-ic","ot-ic","an-tic",
"n-tic","c-tic","at-ic","h-nic","n-ic","m-ic","l-lic","b-lic","-clic","l-ic",
"h-ic","f-ic","d-ic","-bic","a-ic","-mac","i-ac"};

void build_map(hash_map<string,hash_set<string>>& dict)
{
const int n = sizeof(p)/sizeof(char *);
for(int i= 0; i < n; ++i)
{
string line = p[i];
reverse(line.begin(),line.end());
int pos = line.find('-');
dict[line.substr(0,pos)].insert(line.substr(pos + 1,line.le
}

}

string lookup(hash_map<string,hash_set<string> >& dict,string word)
{
string line = word;
reverse(line.begin(),line.end());
int pos = line.find('-');
string ret;

hash_map<string,hash_set<string> >::iterator iter;
if( dict.end() != (iter = dict.find(line.substr(0,pos))))
{
hash_set<string> &h = iter->second;
string temp = line.substr(pos + 1,line.length()-pos-1);
for(int j = 1; j <= (int)temp.length();++j)
{
string c = temp.substr(0,j);
if (h.find(c) != h.end() && c.length() > ret.le
{
ret = c;
}
}

ret = iter->first + "-" + ret;
reverse(ret.begin(),ret.end());

return ret;
}
else
{
cout<<"输入单词不包含字典后缀符"<<endl;
exit(0);
}
}

int main()
{
string sline;
```

```
hash_map<string,hash_set<string> > dict;
build_map(dict);
cout<<"请输入查询单词："<<endl;
while(cin >> sline)
{
cout<< lookup(dict,sline) <<endl;
}

return 0;
}
```

6.编写邮件归并 格式信函发生器

```
[python] view plaincopy;SetupMgrTag
[Unattended]

[GuiUnattended]
AdminPassword=${admin_password}
EncryptedAdminPassword=NO

OEMSkipRegional=1

[UserData]
ComputerName=${computer_name}
```

```
[python] view plaincopyimport os

Template =None
if Template isNone:
t = __import__('Cheetah.Template', globals(), locals(),
['Template'], -1)
Template = t.Template

fp = open('/tmp/unattend.xml','w')
tem = open('/tmp/win03.domain.template').read()
domain_list = {'computer_name':'computer',
'admin_password':'admin',
'Use':'Use'}
info = str(Template(tem, searchList=[domain_list]))
fp.write(info)
fp.close()
```

C++

```
[python] view plaincopyvoid Template(char *fname, char *temp_fname, map<string, st
{
ofstream output(fname);
ifstream temp(temp_fname);
string line;
while (getline(temp, line, '\n'))
{
int bpos =0;
string::size_type pos =0;
while (string::npos != (pos = line.find("${", bpos)))
{
string::size_type epos = line.find("}", pos);
string t = line.substr(pos +2, epos - pos -2);
cout << t << endl;
map<string, string>::iterator iter = dict.find(t);
```

```
            if (dict.end() != iter)
                        {
line.replace(pos, epos - bpos +1, iter->second);
         bpos += iter->second.length();
                        }
                      else
                        {
                bpos = epos +1;
                        }
                        }
             output << line <<"\n\r";
                        }
                 temp.close();
                output.close();
                       }
```

7. 是第二章的改进。

8 2^16 = 65536个数。所以5个七段显示器肯定是够用的。

因为最大的数 65535 有五位数。

```
          #include <iostream>
           #include <memory>
          using namespace std;

        void  showNumber(int i)
                       {
                         int j=i;
                        switch(j)
                            {
                   case 0:printf(" --\n");break;
                     case 1:printf("|");break;
                   case 2:printf("  |\n");break;
                   case 3:printf(" --\n");break;
                     case 4:printf("|");break;
                   case 5:printf("  |\n");break;
                   case 6:printf(" --\n");break;
                          default :break;
                           };
                       }
      void  showNullNumber(int i)
                       {
                         switch(i)
                            {
                    case 0:printf("\n");break;
                     case 1:printf(" ");break;
                   case 2:printf("   \n");break;
                      case 3:printf("");break;
                     case 4:printf(" ");break;
                   case 5:printf("   \n");break;
                    case 6:printf("\n");break;
                         default :break;
                           };
                       }

        void GraphFigure(int i)
```

```
        {
        int show[7];
int show0[]={1,1,1,0,1,1,1};
int show1[]={0,1,0,0,1,0,0};
int show2[]={1,0,1,1,1,0,1};
int show3[]={1,0,1,1,0,1,1};
int show4[]={0,1,1,1,0,1,0};
int show5[]={1,1,0,1,0,1,1};
int show6[]={1,1,0,1,1,1,1};
int show7[]={1,0,1,0,0,1,0};
int show8[]={1,1,1,1,1,1,1};
int show9[]={1,1,1,1,0,1,1};


          switch(i)
              {
case 0:memcpy(show,show0,sizeof(show));break;
case 1:memcpy(show,show1,sizeof(show));break;
case 2:memcpy(show,show2,sizeof(show));break;
case 3:memcpy(show,show3,sizeof(show));break;
case 4:memcpy(show,show4,sizeof(show));break;
case 5:memcpy(show,show5,sizeof(show));break;
case 6:memcpy(show,show6,sizeof(show));break;
case 7:memcpy(show,show7,sizeof(show));break;
case 8:memcpy(show,show8,sizeof(show));break;
case 9:memcpy(show,show9,sizeof(show));break;
            default :break;
             };
     for(int i=0;i<7;++i)
              {
             if(1 == show[i])
                    showNumber(i);
                  else
                 showNullNumber(i);
              }

         }
    int main()
         {
  for(int i=0;i<10;++i)
       {
     GraphFigure(i);
      cout<<"\n\n";

        }
        return 0;
       }
```