# 家谱管理系统

## 函数与结构体.h

#include <iostream>

#include <stdio.h>

#include <fstream>

#include <string>

#include <stdlib.h>

#include<string.h>

#include <time.h>

#include<stack>

using namespace std;

#define Maxx 30

typedef struct

{

int num;//孩子个数

char name[Maxx];///姓名

int sex;//性别

char spouse[Maxx];//配偶的姓名

}DataType;//树数据域的定义

typedef struct Tnode

{

DataType data;

struct Tnode\* nextstr[10];//孩子结点

struct Tnode\* parent;//双亲结点

}Pedtree;//家谱树

void welcome();//主界面

void welcome1();//在添加处的界面

void Create\_tree(Pedtree\*& S, int& k);//用txt文件中的数据构造家谱树

void Create\_Newtree(Pedtree\*& S, int& k);//txt文件的第二个函数

void Create\_tree\_hand(Pedtree\*& S);//直接输出数据构造函数

void Create\_Newtree\_hand(Pedtree\*& ChildrenT);//直接构造的第二个函数

void Display(Pedtree\* S);//输出家谱树

bool Search(Pedtree\* S,char temp[],Pedtree \*&temptree);//查找姓名是否查找符合

void Search1(Pedtree\* S);//递归查找祖先

bool Search2(Pedtree\* S, char temp[], Pedtree\*& temptree);//查找伴侣的姓名是否查找符合

void Search3(Pedtree\* S);//递归查找后代

void Search\_result(Pedtree\* S);//总的查找函数

void add\_newpeople(Pedtree\*& S);//添加主函数

void add\_newchildren(Pedtree\*& S);//添加新孩子函数

void add\_newspouse(Pedtree\*& S);//添加新伴侣函数

void Delete(Pedtree\*& S);//删除主函数

void Deleteall(Pedtree\* &S);//删除孩子部分函数

void New\_File(Pedtree\* S);//存储到文件的函数

## Main.cpp

#include"函数和结构体.h"

char name[Maxx][Maxx], spouse[Maxx][Maxx];//定义全局变量，存储文件中的数据，方便建立家谱树和还原家谱树

int Num[Maxx], Sex[Maxx];

int flag = 0;//标记是否已经构建家谱表

void welcome()//主界面

{

printf("\n\t\t------------------家谱管理-------------------\n\n");

printf("\t\t———————————————————————\n");

printf("\t\t\* \*\n");

printf("\t\t\* 家谱信息管理菜单 \*\n");

printf("\t\t\* \*\n");

printf("\t\t\* 1:输入家谱信息 \*\n");

printf("\t\t\* 2:从文件中读取家谱 \*\n");

printf("\t\t\* 3:查找并输出某人信息 \*\n");

printf("\t\t\* 4:添加新的家庭成员 \*\n");

printf("\t\t\* 5:输出整个家谱信息 \*\n");

printf("\t\t\* 6:删除成员 \*\n");

printf("\t\t\* 7：存储到文件中 \*\n");

printf("\t\t\* 0:退出整个程序 \*\n");

printf("\t\t\* \*\n");

printf("\t\t———————————————————————\n");

}

void welcome1()//添加时的见面

{

printf("\n\n\t\t———————请选择你的操作————————\n\n");

printf("\t —————1.添加某个人的子女的信息——————\n");

printf("\t ——————2.添加某个人的配偶的信息——————\n");

printf("\n\t ——————————3.退出——————————— \n");

}

void Create\_tree\_hand(Pedtree\*& S)//手动输出家谱树

{

char a1[Maxx], a2[Maxx];

int a3, a4;//临时存储文件中的数据

fflush(stdin);//清除键盘缓冲区

system("cls");//清空控制台

printf("请输入第一个人的姓名 伴侣（不存在为0）性别（男为1，女为 0），孩子个数 \n");

scanf("%s %s %d %d", a1, a2, &a3, &a4);//读入

strcpy(S->data.name, a1);//读入到家谱树的总祖先中，先建立总祖先

strcpy(S->data.spouse, a2);

S->data.sex = a3;

S->data.num = a4;//读入到家谱树的总祖先中，先建立总祖先

if (strcmp(S->data.spouse, "0") == 0 && S->data.num != 0)

{

printf("\t\t%s的配偶不存在，不能有孩子！\n", S->data.name);

S->data.num = 0;

}

for (int i = 0; i < S->data.num; i++)//存储孩子部分

{

printf("\n");

Pedtree\* ChildrenT = (Pedtree\*)malloc(sizeof(Pedtree));

S->nextstr[i] = ChildrenT;

printf("请输入%s的 第%d个孩子的 姓名 伴侣（不存在为0）性别（男为1，女为 0），孩子个数 \n",S->data.name,i+1);

scanf("%s %s %d %d", a1, a2, &a3, &a4);//与上同理

strcpy(ChildrenT->data.name, a1);

strcpy(ChildrenT->data.spouse, a2);

ChildrenT->data.sex = a3;

ChildrenT->data.num = a4;

ChildrenT->parent = S;//

if (ChildrenT->data.num != 0)//当孩子的孩子不为0，即有孩子时，先进行孩子的孩子部分输入

Create\_Newtree\_hand(ChildrenT);

}

fflush(stdin);//清除键盘缓冲区

printf("\n\n\t\t- - - - 家谱建造成功- - - - - \n");

}

void Create\_Newtree\_hand(Pedtree\*& S)//构建孩子的孩子，同理

{

char a1[Maxx], a2[Maxx];

int a3, a4;

for (int i = 0; i < S->data.num; i++)

{

printf("\n");

Pedtree\* ChildrenT = (Pedtree\*)malloc(sizeof(Pedtree));

S->nextstr[i] = ChildrenT;

printf("请输入%s的 第%d个孩子的 姓名 伴侣（不存在为0）性别（男为1，女为 0），孩子个数 \n", S->data.name, i + 1);

scanf("%s %s %d %d", a1, a2, &a3,&a4);

strcpy(ChildrenT->data.name, a1);

strcpy(ChildrenT->data.spouse, a2);

ChildrenT->data.sex = a3;

ChildrenT->data.num = a4;

ChildrenT->parent = S;

if (ChildrenT->data.num != 0)

Create\_Newtree\_hand(ChildrenT);

}

}

void Create\_tree(Pedtree\*& S,int &k)//利用文件中的数据构建家谱

{

fflush(stdin);//清除键盘缓冲区

system("cls");

strcpy(S->data.name, name[k]);

strcpy(S->data.spouse, spouse[k]);

S->data.sex = Sex[k];

S->data.num = Num[k];

k++;

if (strcmp(S->data.spouse, "0") == 0 && S->data.num != 0)

{

printf("\t\t%s的配偶不存在，不能有孩子！\n", S->data.name);

S->data.num = 0;

}

for (int i = 0; i < S->data.num; i++)

{

printf("\n");

Pedtree \*ChildrenT = (Pedtree\*)malloc(sizeof(Pedtree));

S->nextstr[i] = ChildrenT;

strcpy(ChildrenT->data.name, name[k]);

strcpy(ChildrenT->data.spouse, spouse[k]);

ChildrenT->data.sex = Sex[k];

ChildrenT->data.num = Num[k];

k++;

ChildrenT->parent = S;

Create\_Newtree(ChildrenT, k);

}

fflush(stdin);//清除键盘缓冲区

printf("\n\n\t\t- - - - 家谱建造成功- - - - - \n");

}

void Create\_Newtree(Pedtree\*& S, int& k)//同理上面

{

for (int i = 0; i < S->data.num; i++)

{

printf("\n");

Pedtree\* ChildrenT = (Pedtree\*)malloc(sizeof(Pedtree));

S->nextstr[i] = ChildrenT;

strcpy(ChildrenT->data.name, name[k]);

strcpy(ChildrenT->data.spouse, spouse[k]);

ChildrenT->data.sex = Sex[k];

ChildrenT->data.num = Num[k];

k++;

ChildrenT->parent = S;

Create\_Newtree(ChildrenT, k);

}

}

void Display(Pedtree\* S)//输出家谱

{

if (strcmp(S->data.name, "0") != 0)//存在这个人

printf("\t\t姓名：%s ", S->data.name);

if (S->data.sex == 1)//男性

{

printf("性别：男\n");

if (strcmp(S->data.spouse, "0") != 0)

printf("\t\t老婆姓名：%s\n", S->data.spouse);

else

printf("\t\t没有老婆，未婚\n");

}

else if (S->data.sex == 0)//女性

{

printf("性别：女\n");

if (strcmp(S->data.spouse, "0") != 0)//是否伴侣

printf("\t\t老公姓名：%s\n", S->data.spouse);

else

printf("\t\t没有老公，未婚\n");

}

for (int i = 0; i < S->data.num; i++)

{

if (strcmp(S->nextstr[i]->data.name, "0") != 0)

printf("\t\t第%d个子女的姓名：%s ", i + 1, (S->nextstr[i])->data.name);

if (((S->nextstr[i])->data.sex) == 1)

printf("性别：男\n");

else if (((S->nextstr[i])->data.sex) == 0)

printf("性别：女\n");

}

cout << "\n";

for (int j = 0; j < S->data.num; j++)//如果有孩子，遍历孩子部分

Display(S->nextstr[j]);

fflush(stdin);//清除键盘缓冲区

}

void Search\_result(Pedtree\* S)

{

Pedtree\* temptree = NULL;//记录找到的位置

char temp[20];

fflush(stdin);//清除键盘缓冲区

system("cls");

printf("\t\t请输入你要查找的人的姓名：");

scanf("%s", temp);

if (Search(S, temp, temptree))//查找名字部分

{

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

if (temptree->parent != NULL)//输出双亲

{

printf("\t\t姓名：%s\t", temptree->data.name);

if (temptree->data.sex == 1)

printf("\t\t性别：男\n");

else if ((temptree->data.sex) == 0)

printf("\t\t性别：女\n");

if ((temptree->parent->data.sex) == 1)

{

printf("\t\t父亲姓名：%s \n", temptree->parent->data.name);

printf("\t\t母亲姓名：%s \n", temptree->parent->data.spouse);

}

else if ((temptree->parent->data.sex) == 0)

{

printf("\t\t父亲姓名：%s \n", temptree->parent->data.spouse);

printf("\t\t母亲姓名：%s \n", temptree->parent->data.name);

}

for (int i = 0; i < temptree->parent->data.num; i++)//输出兄弟部分

{

if (strcmp(temptree->parent->nextstr[i]->data.name, temp) != 0)

printf("\t\t兄弟名称为：%s \n", temptree->parent->nextstr[i]->data.name);

}

if (temptree->parent->parent != NULL)

{

Search1(temptree->parent);//输出祖先部分

}

}

else

{

printf("\t\t第1代\n");//没有双亲，第一代人，总祖先

printf("\t\t姓名：%s \n", temptree->data.name);

if ((temptree->data.sex) == 1)

printf("\t\t性别：男\n");

else if ((temptree->data.sex) == 0)

printf("\t\t性别：女\n");

printf("\t\t无兄弟姐妹 \n");

}

//输出配偶信息

if ((temptree->data.sex) == 1)

{

if (strcmp(temptree->data.spouse, "0") != 0)

printf("\t\t太太姓名：%s\n", temptree->data.spouse);

else

printf("\t\t没有太太，未婚\n");

}

else if ((temptree->data.sex) == 0)

{

if (strcmp(temptree->data.spouse, "0") != 0)

printf("\t\t丈夫姓名：%s\n", temptree->data.spouse);

else

printf("\t\t没有丈夫，未婚\n");

}

if (temptree->data.num != 0)//孩子部分

{

printf("\t\t孩子信息：\n");

for (int j = 0; j < temptree->data.num; j++)

{

if (strcmp(temptree->nextstr[j]->data.name, "0") == 0)

continue;

printf("\t\t\t第%d个孩子：姓名：%s \n", j + 1, temptree->nextstr[j]->data.name);

if ((temptree->nextstr[j]->data.sex) == 1)

printf("\t\t\t\t\t性别：男\n");

else if ((temptree->nextstr[j]->data.sex) == 0)

printf("\t\t\t\t\t性别：女\n");

}

}

for (int i = 0; i < temptree->data.num; i++)//输出后代部分

{

if (temptree->nextstr[i]->data.num != 0)

Search3(temptree->nextstr[i]);

}

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

}

else if (Search2(S, temp, temptree))//如果在伴侣那边，下面同理

{

printf("\n\t\t你要找的人已找到，信息如下：\n");

printf("\t\t姓名：%s \n", temptree->data.spouse);

if ((temptree->data.sex) == 1)

{

printf("\t\t性别：女\n");

printf("\n\t\t无父母信息不存在家谱中,也无兄弟姐妹在家谱中\n");

printf("\n\t\t丈夫姓名：%s\n", temptree->data.name);

}

else if ((temptree->data.sex) == 0)

{

printf("\t\t性别：男\n");

printf("\n\t\t无父母信息不存在家谱中，也无兄弟姐妹在家谱中\n");

printf("\n\t\t太太姓名：%s\n", temptree->data.name);

}

if (temptree->data.num != 0)

{

printf("\t\t孩子信息：\n");

for (int j = 0; j < temptree->data.num; j++)

{

if (strcmp(temptree->nextstr[j]->data.name, "0") == 0)

continue;

printf("\t\t\t第%d个孩子：姓名：%s \n", j + 1, temptree->nextstr[j]->data.name);

if ((temptree->nextstr[j]->data.sex) == 1)

printf("\t\t\t\t\t性别：男\n");

else if ((temptree->nextstr[j]->data.sex) == 0)

printf("\t\t\t\t\t性别：女\n");

}

}

for (int i = 0; i < temptree->data.num; i++)

{

if (temptree->nextstr[i]->data.num != 0)

Search3(temptree->nextstr[i]);

}

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

}

else

{

printf("\t\t\*\*\*该家谱图中没有%s这个人的信息请确认是否输入正确\*\*\*\n\n", temp);

}

}

bool Search(Pedtree\* S, char temp[], Pedtree\*& temptree)//查找名字

{

if (strcmp(S->data.name, temp) == 0)

{

temptree = S;

return true;

}

else

{

for (int i = 0; i < S->data.num; i++)

{

if (Search(S->nextstr[i], temp, temptree))//递归

return true;

}

return false;

}

}

void Search1(Pedtree\* S)//查找祖先

{

if (S->parent != NULL)

{

if (S->parent->data.sex == 1)

printf("\t\t祖先(除双亲外)为 男性 姓名为 %s 配偶为 %s \n", S->parent->data.name, S->parent->data.spouse);

else

printf("\t\t祖先为(除双亲外) 女性 姓名为 %s 配偶为 %s \n", S->parent->data.name, S->parent->data.spouse);

}

if (S->parent->parent != NULL)

Search1(S->parent);

}

bool Search2(Pedtree\* S, char temp[], Pedtree\*& temptree)//查找伴侣部分

{

if (strcmp(S->data.spouse, temp) == 0)

{

temptree = S;

return true;

}

else

{

for (int i = 0; i < S->data.num; i++)

{

if (Search(S->nextstr[i], temp, temptree))

return true;

}

return false;

}

}

void Search3(Pedtree\* S)//查找后代

{

if (S->data.num != 0)

{

for (int i = 0; i < S->data.num;i++)

{

if (strcmp(S->nextstr[i]->data.spouse, "0") != 0)

printf("\t\t后代有%s 配偶为%s\n", S->nextstr[i]->data.name,S->nextstr[i]->data.spouse);

else

printf("\t\t后代有%s 无配偶 \n", S->nextstr[i]->data.name);

}

}

for (int i = 0; i < S->data.num; i++)

{

if (S->nextstr[i]->data.num != 0)

Search3(S->nextstr[i]);

}

}

void add\_newpeople(Pedtree\*& S)//添加新成员

{

fflush(stdin);

int choice;

while (1)

{

fflush(stdin);

system("cls");

welcome1();

cout << "\t 请你选择：";

cin >> choice;

switch (choice)

{

case 1:

{

add\_newchildren(S);//新孩子

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

case 2:

{

add\_newspouse(S);//新伴侣

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

case 3:

return;

}

}

}

void add\_newchildren(Pedtree\*& S)

{

fflush(stdin);

system("cls");

char children\_name[Maxx];

Pedtree\* temptree = NULL;

printf("\n\t\t请输入要添加子女的父母的名字：");//先找到双亲

scanf("%s", children\_name);

if (Search(S, children\_name, temptree) || Search2(S, children\_name, temptree))

{

if (strcmp(temptree->data.spouse, "0") != 0)

{

Pedtree\* Childtree;

int num = temptree->data.num;

Childtree = (Pedtree\*)malloc(sizeof(Pedtree));//创建新的结点

temptree->nextstr[num] = Childtree;

printf("\n\t\t请输入要添加的子女的姓名：");

scanf("%s", Childtree->data.name);

printf("\n\t\t请输入添加子女的性别（1男，0女）：");

fflush(stdin);//清除键盘缓冲区

scanf("%d", &(Childtree->data.sex));

(temptree->data.num)++;

Childtree->parent = temptree;

strcpy(Childtree->data.spouse, "0");

Childtree->data.num = 0;

printf("\n\n\t\t- - - - 子女信息添加成功- - - - - \n");

}

else

printf("\n\t\t配偶不存在，不能添加子女信息\n");

}

else

printf("\n\n\t\t不存在该父母！\n");

fflush(stdin);//清除键盘缓冲区

}

void add\_newspouse(Pedtree\*& S)//与孩子部分同理

{

fflush(stdin);

system("cls");

char pouse\_name[Maxx];

int f = 0;

Pedtree \*temptree=NULL;

printf("\n\t\t请输入要添加人的配偶的名字：");

scanf("%s", pouse\_name);

if (!Search(S, pouse\_name, temptree))

{

printf("\n\n\t\t不存在该人！\n");

f = 1;

}

else

{

if (strcmp(temptree->data.spouse, "0") == 0)

{

printf("\n\t\t请入人的姓名：");

scanf("%s", temptree->data.spouse);

}

else

{

cout << "什么，找小三，我直接哒咩！\n";

f = 1;

}

}

fflush(stdin);

if(f==0)

printf("\n\n\t\t- - - - 配偶信息添加成功- - - - - \n");

}

void Delete(Pedtree\* &S)//删除部分

{

fflush(stdin);//清除键盘缓冲区

system("cls");

char DE\_name[Maxx];

cout << "\t\t请输入要删除的人：";

cin >> DE\_name;

Pedtree\* temptree = NULL;

Pedtree\* head = S;

int i;

if (Search(S, DE\_name, temptree))

{

if (strcmp(temptree->data.name, head->data.name) == 0)

{

flag = 0;

free(S);

}

Deleteall(temptree);

//temptree->parent->data.num--;

for (int i = 0; i < temptree->parent->data.num; i++)//将孩子结点前移，防止报错和反复读入

if (strcmp(temptree->parent->nextstr[i]->data.name, "0") == 0)

{

for (int j = i; j <= temptree->parent->data.num; j++)

temptree->parent->nextstr[j] = temptree->parent->nextstr[j + 1];

temptree->parent->data.num--;//孩子减一

break;

}

cout << "\t\t删除成功\n";

return;

}

else if(Search2(S,DE\_name,temptree))

{

for (i = 0; i < temptree->data.num; i++)

Deleteall(temptree->nextstr[i]);

strcpy(temptree->data.spouse, "0");

temptree->data.num = 0;

printf("\n\t\t删除成功");

}

else

{

printf("\n\n\t\t——————————————————————————\n\n");

printf("\t\t\*\*\*该家谱图中没有%s这个人的信息请确认是否输入正确\*\*\*\n\n", DE\_name);

printf("\t\t——————————————————————————————\n\n");

}

}

void Deleteall(Pedtree\* &S)

{

int i;

if (S)

{

for (i = 0; i < S->data.num; i++)

Deleteall(S->nextstr[i]);

strcpy(S->data.name, "0");

strcpy(S->data.spouse, "0");

S->data.num = 0;

S->data.sex = -1;

//free(S);

}

}

void New\_File(Pedtree\* S)//更新文件

{

FILE\* fp;

stack<Pedtree\*> st;//家谱结构栈，用来临时存储孩子

stack<int> sp;//空格数量栈，为了和结构栈同步，赋给空格方便隔开

Pedtree\* p;

int k = 0;

fp = fopen("Myfamily104.txt", "w");

fprintf(fp, "姓名\t伴侣\t性别\t孩子个数\n");

st.push(S);

sp.push(k);

while (!st.empty())

{

p = st.top();

st.pop();

k = sp.top();

sp.pop();

if (strcmp(p->data.name, "0") != 0)

{

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

fprintf(fp, " ");

fprintf(fp, "%s %s %d %d \n", p->data.name, p->data.spouse, p->data.sex, p->data.num);

if (p->data.num != 0)

{

k = k + 2;//扩大空格

for (int i = 0; i < p->data.num; i++)

{

st.push(p->nextstr[i]);//进栈

sp.push(k);//同步进栈

}

}

}

}

fclose(fp);

}

int main()

{

Pedtree\* S;

char s1[Maxx], s2[Maxx], Garbage[Maxx];

int a, b, s;

FILE\* fp;

//i = 0;

//while (name[i])

//{

// printf("%s", name[i]);

// i++;

//}

//scanf("%d", &i);

int choice, k = 0;

while (1)

{

fflush(stdin);//清除键盘缓冲区

system("cls");

welcome();

printf("\n");

printf("\t\t请您选择:");

cin >> choice;

switch (choice)

{

case 1:

{

if (flag == 0)

{

S = (Pedtree\*)malloc(sizeof(Pedtree));

S->parent = NULL;

Create\_tree\_hand(S);

printf("\t\t构建完成\n");

flag = 1;

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

else

{

cout << "已经建立好了家谱，无需反复" << endl;

int j;

cout << "是否重新构构建 1.是 2.否：";

cin >> j;

switch (j)

{

case 1:

{

Deleteall(S);

flag = 0;

cout << "删除完成\n";

}

case 2:

{

break;

}

}

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

}

case 2:

{

if (flag == 0)

{

k = 0;

fp = fopen("MyFamily104.txt", "r");

if (fp == NULL)

{

printf("can not open file\n");

exit(0);

}

int i = 0;

fscanf(fp, "%s %s %s %s", Garbage, Garbage, Garbage, Garbage);

while (fscanf(fp, "%s %s %d %d\n", s1, s2, &s, &a) != EOF)

{

strcpy(name[i], s1);

strcpy(spouse[i], s2);

Num[i] = a;

Sex[i] = s;

i++;

}

fclose(fp);

S = (Pedtree\*)malloc(sizeof(Pedtree));

S->parent = NULL;

Create\_tree(S, k);

printf("\t\t构建完成\n");

flag = 1;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

else

{

cout << "已经建立好了家谱，无需反复" << endl;

int j;

cout << "是否重新构构建 1.是 2.否：";

cin >> j;

switch (j)

{

case 1:

{

Deleteall(S);

flag = 0;

cout << "删除完成\n";

}

case 2:

{

break;

}

}

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

}

case 3:

{

if (flag == 0)

{

cout << "请先建立族谱" << endl;

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

else

{

fflush(stdin);//清除键盘缓冲区

system("cls");

Search\_result(S);

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

}

case 4:

{

if (flag == 0)

{

cout << "请先建立族谱" << endl;

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

else

{

fflush(stdin);//清除键盘缓冲区

system("cls");

add\_newpeople(S);

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

}

case 5:

{

if (flag == 0)

{

cout << "请先建立族谱" << endl;

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

else

{

fflush(stdin);//清除键盘缓冲区

system("cls");

printf("\n\n\t\t整个家谱主要的信息如下：\n");

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

Display(S);

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

}

case 6:

{

if (flag == 0)

{

cout << "请先建立族谱" << endl;

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

else

{

Delete(S);

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

}

case 7:

{

if (flag == 0)

{

cout << "请先建立族谱" << endl;

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

else

{

fflush(stdin);

system("cls");

New\_File(S);

cout << "存储完成\n";

int i;

printf("\t\t输出0返回");

scanf("%d", &i);

break;

}

}

case 0:

{

cout << "\t\t感谢使用本小程序！" << endl;

return 0;

}

default:

cout << "\t\t输出不正确，再试试吧！" << endl;

break;

}

}

}

# 停车场管理系统

## 函数定义和结构体定义.h

#pragma once

#include"函数定义和结构体定义.h"

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <string.h>

#include <cstring>

#include <time.h>

using namespace std;

//头文件部分

#define Price 0.1//每一秒的价格

#define MAX\_car 5//停车场最大容量

#define MAX\_wait 4//候车场最大容量

typedef struct

{

int in\_time;//进场时间

int out\_time;//出场时间

char ct[100];//停车时间信息

char License\_plate[20];//车牌号

}Car;

typedef struct node

{

Car stop[MAX\_car];

int top;

}car\_stop;//停车栈

typedef struct //候车场

{

Car wait[MAX\_wait];

int count;

int front, rear;

}Waitting;//候车循环队列

void Car\_Leave(car\_stop& G, Waitting& P);//停车场出车

void Display(car\_stop G);//显示停车场车辆信息

void Display\_wait(Waitting P);//显示候车场车辆信息

void P\_Leave(Waitting& P);//候车场出车

void car\_P\_come(Waitting& P, char C[]);//候车场进车

void car\_come(car\_stop& G, Waitting& P);//停车场进车

void search(car\_stop G, Waitting P);//查找车辆

void welcome(car\_stop& G, Waitting& P);//菜单界面

## Main.cpp

#include"函数定义和结构体定义.h"

void Display(car\_stop G)

{

int i = G.top;

if (i == -1)

cout << "目前停车站为空" << endl;//当停车场为空时

else

{

time\_t t1;

long int t = time(&t1);

printf("\t车牌号\t\t停放时间\t当前所需支付金额\t停放位序\n");

while (i != -1)

{

printf("\t%s\t\t%d时%d分\t\t%d元\t\t\t第%d个车位\n",

G.stop[i].License\_plate, (t - G.stop[i].in\_time) / 360, (t - G.stop[i].in\_time) / 60, (((int )(t - G.stop[i].in\_time)/60+1)\*(6)), i + 1);

//停车时间 为当前时间-进车时间

i--;

}

}

}

void Display\_wait(Waitting P)

{

int i = P.front;

int k = 1;

if(P.count==0)

printf("便道目前为空\n");

else

printf("\t车牌号\t\t停放位序\n");

while (i != P.rear && k <= P.count) //用读取的数量来作为判断的标准

{

printf("\t%s\t\t第%d个\n", P.wait[i].License\_plate, k);

i = (i + 1) % MAX\_wait;

k++;

}

}

void car\_P\_come(Waitting& P,char C[])//当停车场满后会进行这个操作

{

int i = P.front,c=0;

if(P.count>0&&(P.front==P.rear+1)%MAX\_wait)

printf("便道已满，请下次再来！\n");

else

{

while (c < P.count)//判断候车场是否有该车

{

if (strcmp(P.wait[i].License\_plate, C) == 0)

{

printf("输入有误，此汽车已存在！\n");

return;

}

i--;

c++;

}

strcpy(P.wait[P.rear].License\_plate, C);//进队操作

P.rear = (P.rear + 1) % MAX\_wait;

P.count++;

printf("牌照为%s的汽车停入便道上\n", C);

}

}

void car\_come(car\_stop& G, Waitting& P)

{

char C[20];

printf("请输入即将停车的车牌号：");

getchar();

gets\_s(C); //vs特性

int k;

k = strlen("闽D 12345");

if (strlen(C) != k)

{

printf("输入错误！！！\n");

}

else

{

int i = G.top;

while (i != -1) {

if (strcmp(G.stop[i].License\_plate, C) == 0) { //判断停车场是否存在当前车辆

printf("输入有误，此汽车已存在！\n");

return;

}

i--;

}

if (G.top >= MAX\_car - 1)

{

printf("停车场已满\n请转到候车场\n");

car\_P\_come(P, C);

}

else

{

time\_t t1;

long int t = time(&t1); //时间

char\* t2;

t2 = ctime(&t1); //当前时间

G.stop[++G.top].in\_time = t;

strcpy(G.stop[G.top].ct, t2);//复制信息到G中

strcpy(G.stop[G.top].License\_plate, C);

printf("牌照为%s的汽车停入停车位的第%d车位，当前时间：%s\n", C, G.top + 1, t2);

}

}

printf("\n\t\t\t是否继续进行进车？是：1，否：2\n");//判断是否进行进车

int c;

cin >> c;

if (c == 1)

car\_come(G, P);

else if (c == 2)

return;

else

printf("输入不对，再试试吧！\n");

}

void Car\_Leave(car\_stop& G,Waitting &P)

{

char C[20];

printf("请输入即将离开的车牌号：");

getchar();//清除输出的回车，防止被gets读取(以下同理）

gets\_s(C);//VS特定识别符

int i,flag=1;

if (G.top >= 0)

{

for (i = G.top; i >= 0; i--)

{

if (strcmp(G.stop[i].License\_plate, C) == 0)//寻找要出的车辆

{

flag = 0;//标记位

break;

}

}

}

if (flag == 0)//找到

{

Car temp;

int j = i;

temp = G.stop[i];

while (j <= G.top&&j>-1)

{

G.stop[j] = G.stop[j + 1];//将车辆前移

j++;

}

G.top--;

time\_t t1;//利用系统自带的时间函数计算出入时间

long int t = time(&t1);//

temp.out\_time = t;

char\* t2;

t2 = ctime(&t1);//获得当前时间

printf("离开时间%s\n需付费%d元\n", t2, (((int)(t - G.stop[i].in\_time) / 60 + 1) \*6) );

while (G.top < MAX\_car - 1)//如果候车场不为空时，进入停车场

{

if (P.count == 0)

break;

else

{

strcpy(G.stop[++G.top].License\_plate, P.wait[P.front].License\_plate);

printf("牌照为%s的汽车从便道中进入停车位的%d车位\n", P.wait[P.front].License\_plate, G.top + 1);

time\_t t1;

long int t = time(&t1);

char\* t2;

G.stop[G.top].in\_time = t;

P.front = (P.front + 1) % MAX\_wait;//循环队列

P.count--;

}

}

}

else

printf("停车场内没有该汽车的信息！\n");

}

void P\_Leave( Waitting& P)//

{

int i,flag=1;

char C[20];

printf("请输入即将离开的车牌号：");

getchar();

gets\_s(C);

if (P.count <= 0)

{

printf("便道上不存在汽车!\n");

return;

}

i = P.front;

int c = 0;

while (c<P.count)

{

if (strcmp(P.wait[i].License\_plate, C) == 0)//寻找相同的车牌号

{

flag = 0;

break;

}

i++;

c++;

}

if (flag == 0)

{

printf("\n牌照为%s的汽车离开候车场，不收取任何费用！\n", P.wait[i].License\_plate);

c = 0;

int j = i;

int temp = (P.rear - i + MAX\_wait) % MAX\_wait;//计算出从i(找到的队列下标)到最后一个队列有几个元素，作为判断标准

while (c<temp)

{

P.wait[j] = P.wait[((j + 1) % MAX\_wait)];//候车站队列前移

j++;

c++;

}

P.count--;

}

}

void search(car\_stop G, Waitting P)

{

char C[20];

printf("请输入要搜索的车牌号：\n");

getchar();

gets\_s(C);

int i,flag=0,k,j;

time\_t t1;

long int t = time(&t1);

if(G.top>=0)

for (i = G.top; i >=0; i--)

{

if (strcmp(G.stop[i].License\_plate, C) == 0)

{

printf("此汽车在停车场内，信息如下：\n");

printf("\t车牌号\t\t停车位序\t当前所需支付金额\t进入时间\t\n");

printf("\t%s\t\t第%d个\t\t%d元\t\t\t%s", G.stop[i].License\_plate, i + 1, (((int)(t - G.stop[i].in\_time) / 60 + 1) \* 6), G.stop[i].ct);

flag = 1;

break;

}

}

if (flag == 0 && P.count > 0)//停车场没有，候车场不为空时

{

i = P.front, k = 1, j = P.rear;

while (i != j)

{

if (strcmp(P.wait[i].License\_plate, C) == 0)

{

printf("此汽车在停便道上\n");

printf("\t车牌号\t\t停车位序\n");

printf("\t%s\t第%d个", P.wait[i].License\_plate, k);

flag = 1;

break;

}

i++;

k++;

}

}

if (flag==0)

printf("停车场内外不存在该汽车信息！\n");

}

void welcome(car\_stop &G,Waitting &P)

{

printf("\t————————————目前停车场状况—————————————\n");

printf("\t停车场共有%d个车位,当前停车场共有%d辆车,等候区共有%d辆车\n", MAX\_car, G.top + 1, (P.rear - P.front + MAX\_wait) % MAX\_wait);

printf("\t——————————————————————————————\n");

printf("\t| 1.停车场停车信息 |\n");

printf("\t| 2.便道上停车信息 |\n");

printf("\t| 3.汽车到达停车场 |\n");

printf("\t| 4.汽车离去停车场 |\n");

printf("\t| 5.汽车离去候车场 |\n");

printf("\t| 6.查找汽车功能　 |\n");

printf("\t| 7.退出管理系统 |\n");

printf("\t|收费标准：本停车场按照%.0f元/分钟计费，望周知(时间向上取整)|\n", 60 \* Price);

printf("\t| |\n");

printf("\t————————————————————————————————\n");

}

int main()

{

int flag=1;

car\_stop G;//

G.top = -1;//初始化栈

Waitting P;

P.rear = P.front = 0;//初始化队列

P.count = 0;

while (flag)

{

system("cls");//清空当前界面

welcome(G, P);

int i;

cin >> i;

switch (i)

{

case 1:

{

Display(G);

break;

}

case 2:

{

Display\_wait(P);

break;

}

case 3:

{

car\_come(G, P);

break;

}

case 4:

{

Car\_Leave(G, P);

break;

}

case 5:

{

P\_Leave(P);

break;

}

case 6:

{

search(G, P);

break;

}

case 7:

{

printf("\n欢迎您再次使用本系统呦 \n\n");

flag = 0;

return 0;

break;

}

default:

break;

}

while (1)

{

printf("\n输出0返回\n");

cin >> i;

if (i == 0)

break;

else

printf("您的输入有误，请重新输入\n");

}

}

}