计算机学院本科实验教学文档管理规范

**文件名统一格式为：实验名称-实验X-班级-学号-姓名**（附件2）

**实 验 报 告**

**课程名称：** c++程序设计

**实验项目：**预留地开发项目

**实验仪器：** QT

**学 院： 计算机学院**

**专 业： 计算机大类**

**班级姓名： 计类1711 蒋志文**

**学 号： 2017010736**

**日 期： 2018.10.25**

**同组成员：计类1711张宜放 2017012204**

**计类1709杨震栋 2017011436**

**计类1709 姚江 2017011439**

# 实验目的

1.完成预留地规划方案生成软件

2.掌握如何实现界面

3.掌握继承派生、虚拟和多态

4.掌握输入输出流的编写，生成文件储存方案规划结果。

5.学会对自己的软件写功能说明

6.学会对自己的函数进行逐个测试

7.实现团队合作、分工完成任务

# 实验内容

**1.要求：**

实现预留地规划方案生成软件。要求有界面，生成文件储存方案规划结果。需要对自己的软件写功能说明，对自己的函数进行测试。

**2.功能：**

根据用户输入的场地面积和总预算，若预算充足则输出方案的草图，在草图旁边输出合理方案的具体内容，并在系统内部将方案生成的结果写入文件中；预算不足则提示信息，为何无法生成方案。

如果本次输入数据与上一次方案的输入数据相同时（即用户多次点击生成键），结果将不会被重复写入文件。

**3.使用流程：**

输入：

①场地的面积，输入的范围为：0 ~ 99999999.999999；

②预计的预算，输入的范围为：0 ~ 99999999.999999；

③是否将未利用部分用草坪填满，用户在单选框中选择；

输出：

界面上用户只需要按要求输入数据，点击生成方案按钮，就能将内部算法函数执行的返回结果显示到界面上。

**4.类的介绍**

**Shape类：**

**1.路径：**

定义于 /Headers/base.h 与 /Sources/base.cpp，主要内容均写于base.h。

**2.描述：**

包含了纯虚函数，是抽象类，是接下来所有几何形状的父类。

定义了纯虚方法area, 其中const关键字表明了这个函数不能修改其数据成员且可以被常量对象访问, =0表示了这个函数是纯虚函数。

因为价格基本都是基于面积，所以定义这个纯虚函数要求之后的几何形状类都需要计算面积。

﻿**Circular类：**

**1.路径：**

成员变量和函数定义于﻿circular.h中，函数实现于﻿circular.cpp中。

**2.描述**

该类继承了Shape类，为圆形类，用于表示游泳池；

成员变量：radius：存储了圆形的半径，类型为double。

函数：

area：返回一个double类型的值，无参数，利用圆形面积公式 S = π\* r^2 ，计算圆形的面积并返回；

set\_radius：返回类型为void，有一个double类型的参数，修改类的成员变量radius；

get\_perimeter：返回一个double类型的值，无参数，利用圆形周长公式C = 2 \* π\* r，计算圆形的周长并返回；

get\_radius：返回一个double类型的值，无参数，返回半径即成员变量radius的值。

﻿**Rectangle类：**

**1.路径**

成员变量和函数定义于﻿rectangle.h中，函数实现于﻿rectangle.cpp中。

**2.描述**

该类继承了Shape类，为矩形类，用于表示休息区。

成员变量：

length：存储矩形的长，类型为double；

width：存储矩形的宽，类型为double。

函数：

﻿area：返回一个double类型的值，无参数，利用矩形面积公式 S = length \* width，计算矩形的面积并返回；

﻿set\_attribute：返回类型为void，两个double类型的参数，用于修改类的成员变量length和 width, 第一个参数对应修改length 第二个参数对应修改 width；

﻿get\_perimeter：返回一个double类型的值，无参数，利用矩形周长公式 C = 2 \* (length + width)，计算矩形的周长并返回；

get\_length：返回一个double类型的值，无参数，返回矩形的长即成员变量length的值；

get\_width：返回一个double类型的值，无参数，返回矩形的宽即成员变量width的值。

**Trapezium类：**

**1.路径**

1、成员变量和函数定义于﻿trapezium.h中；

2、函数实现于﻿trapezium.cpp中。

**2.描述**

该类继承了Shape类，为梯形类，且梯形为等腰梯形，且较长的下底的底角为45度，用于表示休息区。

成员变量：

upLength：存储梯形的上底，类型为double ；

downLength：存储梯形的下底，类型为double ；

height：存储梯形的高，类型为double 。

函数：

area：返回一个double类型的值，无参数，利用梯形面积公式 S = 1/ 2 \* height \* (upLength + downLength)，计算梯形的面积并返回；

set\_attribute：返回类型为void，一个double类型的参数，用于修改类的成员变量downLength；

get\_perimeter：返回一个double类型的值，无参数，利用等腰梯形周长公式 C =上底 + 下底 + 2 \* 腰长，计算梯形的周长并返回；

get\_downLength：返回一个double类型的值，无参数，返回梯形的下底即成员变量downLength的值；

get\_upLength：返回一个double类型的值，无参数，返回梯形的上底即成员变量upLength的值；

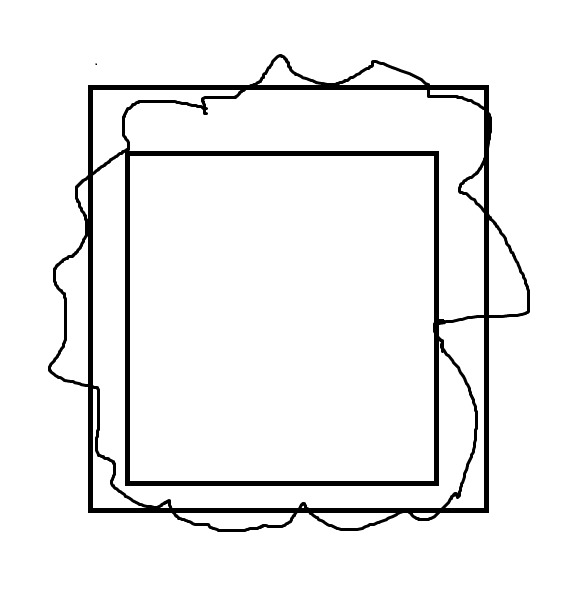
get\_height：返回一个double类型的值，无参数，返回梯形的高即成员变量height的值。

**5.核心算法详细介绍**

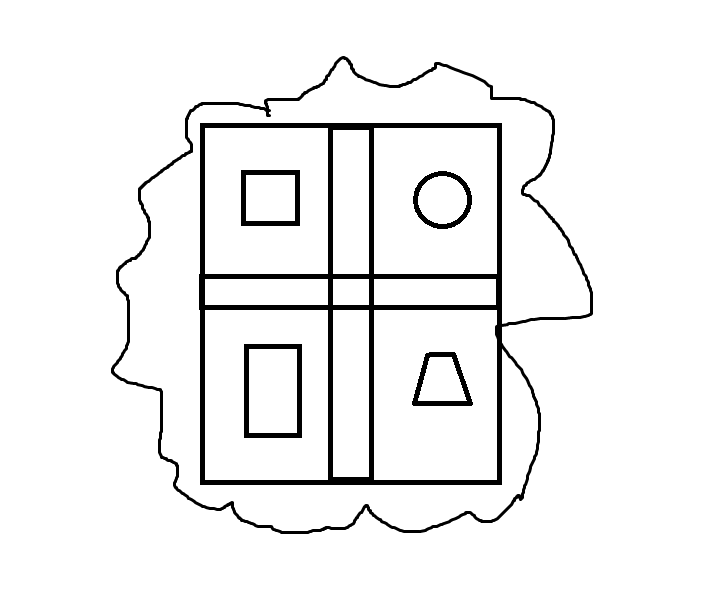
**主要算法包括两个逻辑函数，calculate和search**

第一，将预留地做合理化处理。用户输入总场地面积开方向下取整，再去掉得到边长的1/20以减小误差实现场地的最大利用（图形不规则，内部画方可能有边角取不到，消去一点以合理化）将不规则场地化作规则正方形。

如下图所示，场地不规则，外方形为直接开方所得，内方形为经算法处理实际可利用面积。



第二，search函数寻找最合理方案。首先将场地化为四块，在这四块分别建设不同设施，有圆形泳池（外围有栅栏）、矩形更衣室、方形超市和梯形休息区，中间过道（两边均有栅栏）默认为三米宽；再利用第一步中我们得到可利用边长，用可利用边长千分之一作为计数器，四部分边长（或某参数）从零开始寻找合理预算能建成的最大面积。以千分之一作为计数器是为了减少计算时间的同时保证精度，若以1米为计量单位对10米场地计算误差过大，浪费空间过大；对10000米场地来说计算耗时过多。选取合理计数器以边长的千分之一为单位。下图为方案草图



在这一步如果最终计算出合理边长不足可以利用边长的1/4（即分块变长的1/2），说明规划完成方案场地（过道加设施）利用率低于50%，则输出预算不足无法形成方案。

第三，可处理未利用场地，以实现利用最大化。外围因不规则截去场地设计方案为种树，绿色森林环绕着整个娱乐场；内部（圆形、梯形、矩形边角）未利用场地由客户选择是否以草坪覆盖，无论是否填充都可以给客户输出最合理的方案和各部分支出预算。当然，覆盖草坪花费可能会导致其他部分预算减少，我们通过压缩草地花费、总场地四部分平均减少预算，以尽可能减小铺盖草坪对其他部分场地建设的影响。

最后，以字符串输出各类信息：各部分单价，预计方案造价，总预算以及场地利用情况。

**算法参数列表**

|  |  |  |  |
| --- | --- | --- | --- |
| 变量 | 类型 | 解释 | 其他注释 |
| area | double | 场地面积 | 传参、用户输入 |
| budget | double | 总预算金额 | 传参、用户输入 |
| use\_budget | double | 终预算 | 引用值 |
| side | double | 边长 | 引用值，各形状最终参数 |
| with\_grass | bool | 是否要草坪 | 1则草地填满剩余区域，0反之 |
|  |  |  | 以上为calculate函数所需参数 |
|  |  |  |  |
| square\_side | double | 外边长 | 最大可利用方形边长 |
| count | double | 计数器 | 千分之一外边长，扩展最小单位 |
| tree\_budget | double | 树所需预算 | 必须满足，无法减少budget |
| result | string | 最终方案 | 为文字描述 |
|  |  |  | 以上为calculate函数内部定义参数 |
|  |  |  |  |
| area | 同上 |  |  |
| budget | 同上 |  |  |
| square\_side | 同上 |  |  |
| count | 同上 |  |  |
| tree\_budget | 同上 |  |  |
| use\_budget | 同上 |  | 依然作为引用 |
| side | 同上 |  | 依然作为引用 |
|  |  |  | 以上为search函数所需参数 |
|  |  |  |  |
| square\_shop | Rectangle | 方形超市 |  |
| rectangle\_locker | Rectangle | 矩形更衣室 |  |
| circular\_poor | Circular | 圆形泳池 |  |
| trapezoid\_rest | Trapezoid | 梯形休息区 |  |
| result | string | 最终方案 | 为文字描述 |
|  |  |  |  |
| shop\_area | double | 超市面积 | 用于计算造价 |
| locker\_area | double | 更衣室面积 | 用于计算造价 |
| poor\_area | double | 泳池面积 | 用于计算造价 |
| fence\_length | double | 篱笆长度 | 用于计算造价 |
| reat\_area | double | 休息区面积 | 用于计算造价 |
|  |  |  |  |
| shop\_budget | double | 超市预算 |  |
| locker\_budget | double | 更衣室预算 |  |
| poor\_budget | double | 泳池预算 |  |
| fence\_budget | double | 篱笆预算 |  |
| reat\_budget | double | 休息区预算 |  |
|  |  |  | 以上为search函数自定义参数 |
|  |  |  |  |
| grass\_price |  | 草造价 | All |
| tree\_price |  | 树造价 | 元/平方米 |
| shop\_price |  | 超市造价 | 元/平方米 |
| locker\_price |  | 更衣室造价 | 元/平方米 |
| poor\_price |  | 泳池造价 | 元/平方米 |
| fence\_price |  | 篱笆造价 | 元/米 |
| reat\_price |  | 休息区造价 | 元/平方米 |

**6.分工协作**

蒋志文 界面

张宜放 算法

杨震东 类

姚江 测试

# 实验过程

**源代码：**

**1. base.cpp**

#include "base.h" //放入公共的操作，操作的实现在.h文件中,本次实验未启用

**2.base.h**

#ifndef MANAGER\_H

#define MANAGER\_H

#define PI 3.14

class Shape {

public:

virtual double ***area***() const = 0;

};

#endif // MANAGER\_H

**3. calculate.h**

#ifndef CALCULATE\_H

#define CALCULATE\_H

#include <iostream>

#include <string.h>

#include <math.h>

#include <utils.h>

#include <QString>

#define tree\_price 35

#define grass\_price 1000

#define shop\_price 85

#define locker\_price 45

#define poor\_price 25

#define fence\_price 15

#define reat\_price 55

QString **calculate**(double area, double budget, double &use\_budget, double &side, bool with\_grass){ //参数:场地面积,预算,(引用传回)使用金额

double square\_side, count, tree\_budget;

QString result, result\_1;

square\_side = floor(sqrt(area));

square\_side = square\_side - 1/20 \* square\_side; //切割不规则图形,以向下取整得到方形,因边缘不规则边长去5%以消误差

tree\_budget = tree\_price \*(area - square\_side \* square\_side); //剩余面积种树填满

count = square\_side / 1000; //计数器,千分之一边长作为扩展最小单位

if(with\_grass){

budget -= grass\_price;

result.append("草坪预计花费");

result.append(QString("%1").arg(grass\_price));

result.append("元;\n");

result\_1 = search(area, budget, square\_side, count, tree\_budget, use\_budget, side, with\_grass);

result.append(result\_1);

}

else{

result = search(area, budget, square\_side, count, tree\_budget, use\_budget, side, with\_grass);

}

return result;

}

#endif // CALCULATE\_H

**4.** **circular.cpp**

#include "circular.h"

double Circular::***area***() const {

return PI \* radius \* radius;

}

void Circular::**set\_radius**(double r) {

radius = r;

}

double Circular::**get\_perimeter**() {

return 2 \* PI \* radius;

}

double Circular::**get\_radius**() {

return radius;

}

**5.** **circular.h**

#ifndef CIRCULAR\_H

#define CIRCULAR\_H

#include "base.h"

class Circular : public Shape {

public:

virtual double ***area***() const;

void **set\_radius**(double r);

double **get\_perimeter**();

double **get\_radius**();

protected:

double radius;

};

#endif // CIRCULAR\_H

**6.** **design\_manger.pro**

QT += core gui

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

TARGET = design\_manger

TEMPLATE = app

DEFINES += QT\_DEPRECATED\_WARNINGS

CONFIG += c++11

SOURCES += \

main.cpp \

mymainwindow.cpp \

circular.cpp \

rectangle.cpp \

base.cpp \

trapezium.cpp

HEADERS += \

mymainwindow.h \

circular.h \

rectangle.h \

base.h \

utils.h \

calculate.h \

trapezium.h

FORMS += \

mymainwindow.ui

# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

DISTFILES += \

C:/Users/jzw/Desktop/矩形方案.png \

C:/Users/jzw/Desktop/圆形方案.png

RESOURCES += \

res.qrc

**7.** **mymainwindow.cpp**

#include "mymainwindow.h"

#include "ui\_mymainwindow.h"

#include "circular.h"

#include "rectangle.h"

#include "math.h"

#include <QString>

#include <QDebug>

#include <QButtonGroup>

#include <QPushButton>

#include <calculate.h>

#include <QMessageBox>

#include <QFile>

double pre\_budget = 0;

double pre\_area = 0;

bool same = false;

int pre\_btn\_type = -1;

MyMainWindow::**MyMainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MyMainWindow)

{

ui->setupUi(this);

this->setWindowTitle("方案生成器");

}

MyMainWindow::~MyMainWindow()

{

delete ui;

}

void MyMainWindow::**on\_btn\_1\_clicked**()

{

double budget = ui->input\_1->text().toDouble();

double area = ui->input\_2->text().toDouble();

double use\_budget = 0;

double side = 0;

int flag = 0;

QString plan, plan\_res;

if (pre\_budget == budget and pre\_area == area){

same = true;

}else

same = false;

QButtonGroup \*grp=new QButtonGroup(this);

grp->addButton(ui->type1);

grp->addButton(ui->type2);

ui->label\_title->setText(QString("方案草图"));

ui->label\_title\_2->setText(QString("方案详情"));

ui->label\_image->setPixmap(QPixmap(":/image/img/plan.jpg"));

ui->label\_image->setScaledContents(true);

if (ui->type2 == grp->checkedButton())

{

flag = 0;

// 初始化

if (pre\_btn\_type == -1){

pre\_btn\_type = 0;

}

}

else if (ui->type1 == grp->checkedButton()) {

flag = 1;

if (pre\_btn\_type == -1){

pre\_btn\_type = 1;

}

}

plan = QString("树造价：%1元/平方米\n").arg(tree\_price);

plan.append(QString("树造价：%1元/平方米\n").arg(tree\_price));

plan.append(QString("超市造价：%1元/平方米\n").arg(shop\_price));

plan.append(QString("更衣室造价：%1元/平方米\n").arg(locker\_price));

plan.append(QString("泳池造价：%1元/平方米\n").arg(poor\_price));

plan.append(QString("过道造价：%1元/米\n").arg(fence\_price));

plan.append(QString("休息区造价：%1元/平方米\n").arg(reat\_price));

plan.append(QString("草总造价：%1元\n").arg(grass\_price));

plan\_res = calculate(area, budget, use\_budget, side, flag);

plan.append(plan\_res);

this->ui->res->setText(plan);

if (!same or pre\_btn\_type != flag){

// 将方案写入文件

QString filename = "res.txt"; //当前文件夹中

QFile file(filename);

if(!file.open(QIODevice::ReadWrite|QIODevice::Text | QIODevice::Append))

{

QMessageBox::warning(this,"file write","can't open",QMessageBox::Yes);

}

QTextStream in(&file);

in << plan << endl;

}

// 更新全局变量的值

pre\_budget = budget;

pre\_area = area;

pre\_btn\_type = flag;

}

**8.** **mymainwindow.h**

#ifndef MYMAINWINDOW\_H

#define MYMAINWINDOW\_H

#include <QMainWindow>

namespace Ui {

class MyMainWindow;

}

class MyMainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MyMainWindow(QWidget \*parent = nullptr);

~MyMainWindow();

private slots:

void **on\_generate\_button\_pressed**();

void **on\_btn\_1\_clicked**();

private:

Ui::MyMainWindow \*ui;

};

#endif // MYMAINWINDOW\_H

**9.** **rectangle.cpp**

#include "rectangle.h"

double Rectangle::***area***() const{

return this->length \* this->width;

};

void Rectangle::**set\_attribute**(double l, double w) {

length = l;

width = w;

}

double Rectangle::**get\_perimeter**() {

return 2 \* (length + width);

}

double Rectangle::**get\_length**(){

return length;

}

double Rectangle::**get\_width**() {

return width;

}

**10.** **rectangle.h**

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

#include "base.h"

class Rectangle : public Shape {

public:

virtual double ***area***() const;

void **set\_attribute**(double l, double w);

double **get\_perimeter**();

double **get\_length**();

double **get\_width**();

protected:

double length;

double width;

};

#endif // RECTANGLE\_H

**11.** **trapezium.cpp**

#include "Trapezium.h"

#include <cmath>

#include <iostream>

using namespace std;

Trapezium::**Trapezium**() {

upLength = 0;

downLength = 0;

height = 0;

}

Trapezium::**Trapezium**(double downLength) {

this->upLength = downLength / 3;

this->downLength = downLength;

this->height = downLength / 3;

}

double Trapezium::***area***() const {

return (upLength + downLength) \* height / 2;

}

double Trapezium::**get\_perimeter**() {

double h;

h = sqrt((upLength - downLength / 2) \* (upLength - downLength / 2) + height \* height);

return h \* 2 + upLength + downLength;

}

double Trapezium::**get\_downLength**() {

return this->downLength;

}

double Trapezium::**get\_upLength**() {

return this->upLength;

}

double Trapezium::**get\_height**() {

return this->height;

}

void Trapezium::**set\_attribute**(double downLength) {

this->upLength = downLength / 3;

this->downLength = downLength;

this->height = downLength / 3;

}

**12.** **trapezium.h**

#ifndef TRAPEZOID\_Trapezium H

#define TRAPEZOID\_Trapezium H

#include "base.h"

class Trapezium : public Shape {

public:

virtual double ***area***() const;

void **set\_attribute**(double downLength);

double **get\_perimeter**();

double **get\_upLength**();

double **get\_downLength**();

double **get\_height**();

Trapezium();

Trapezium(double downLength);

protected:

double upLength;

double downLength;

double height;

};

#endif //TRAPEZOID\_Trapezium H

**13.** **utils.h**

#ifndef UTILS\_H

#define UTILS\_H

#include <iostream>

using namespace std;

#include <math.h>

#include <QString>

#include <rectangle.h>

#include <circular.h>

#include <trapezium.h>

#define tree\_price 35

#define grass\_price 1000

#define shop\_price 85

#define locker\_price 45

#define poor\_price 25

#define fence\_price 15

#define reat\_price 55

QString **search**(double area, double budget, double square\_side, double count, double tree\_budget, double &use\_budget, double &side, bool grass){

Rectangle square\_shop, rectangle\_locker;

Circular circular\_poor;

Trapezium trapezoid\_rest;

QString result;

double shop\_area, locker\_area, poor\_area, fence\_length, reat\_area;

double shop\_budget, locker\_budget, poor\_budget, fence\_budget, reat\_budget;

square\_side -= 3;

for(; (use\_budget < budget) && (side < square\_side / 2); side += count){

square\_shop.set\_attribute(side, side);

rectangle\_locker.set\_attribute(side, side \* 0.75);

circular\_poor.set\_radius(side / 2);

trapezoid\_rest.set\_attribute(side);

shop\_area = square\_shop.area();

locker\_area = rectangle\_locker.area();

poor\_area = circular\_poor.area();

fence\_length = circular\_poor.get\_perimeter();

reat\_area = trapezoid\_rest.area();

shop\_budget = shop\_area \* shop\_price;

locker\_budget = locker\_area \* locker\_price;

poor\_budget = poor\_area \* poor\_price;

fence\_budget = (fence\_length + 4 \* square\_side)\* fence\_price;

reat\_budget = reat\_area \* reat\_price;

use\_budget = tree\_budget + shop\_budget + locker\_budget + poor\_budget + fence\_budget + reat\_budget;

}

side -= count;

if(side < square\_side / 4){

result = "预算过少,合理方案场地利用率不足50%,无法建设\n";

return result;

}

result.append("设计方案总预算");

if(grass)

use\_budget += grass\_price;

result.append(QString("%1").arg(use\_budget));

result.append("元。\n其中，外围场地利用（种树）花费");

result.append(QString("%1").arg(tree\_budget));

result.append("元;\n泳池建设花费");

result.append(QString("%1").arg(poor\_budget));

result.append("元;\n过道花费");

result.append(QString("%1").arg(fence\_budget));

result.append("元;\n商店建设花费");

result.append(QString("%1").arg(shop\_budget));

result.append("元;\n更衣室建设花费");

result.append(QString("%1").arg(locker\_budget));

result.append("元;\n休息区建设花费");

result.append(QString("%1").arg(reat\_budget));

result.append("元;\n");

return result;

}

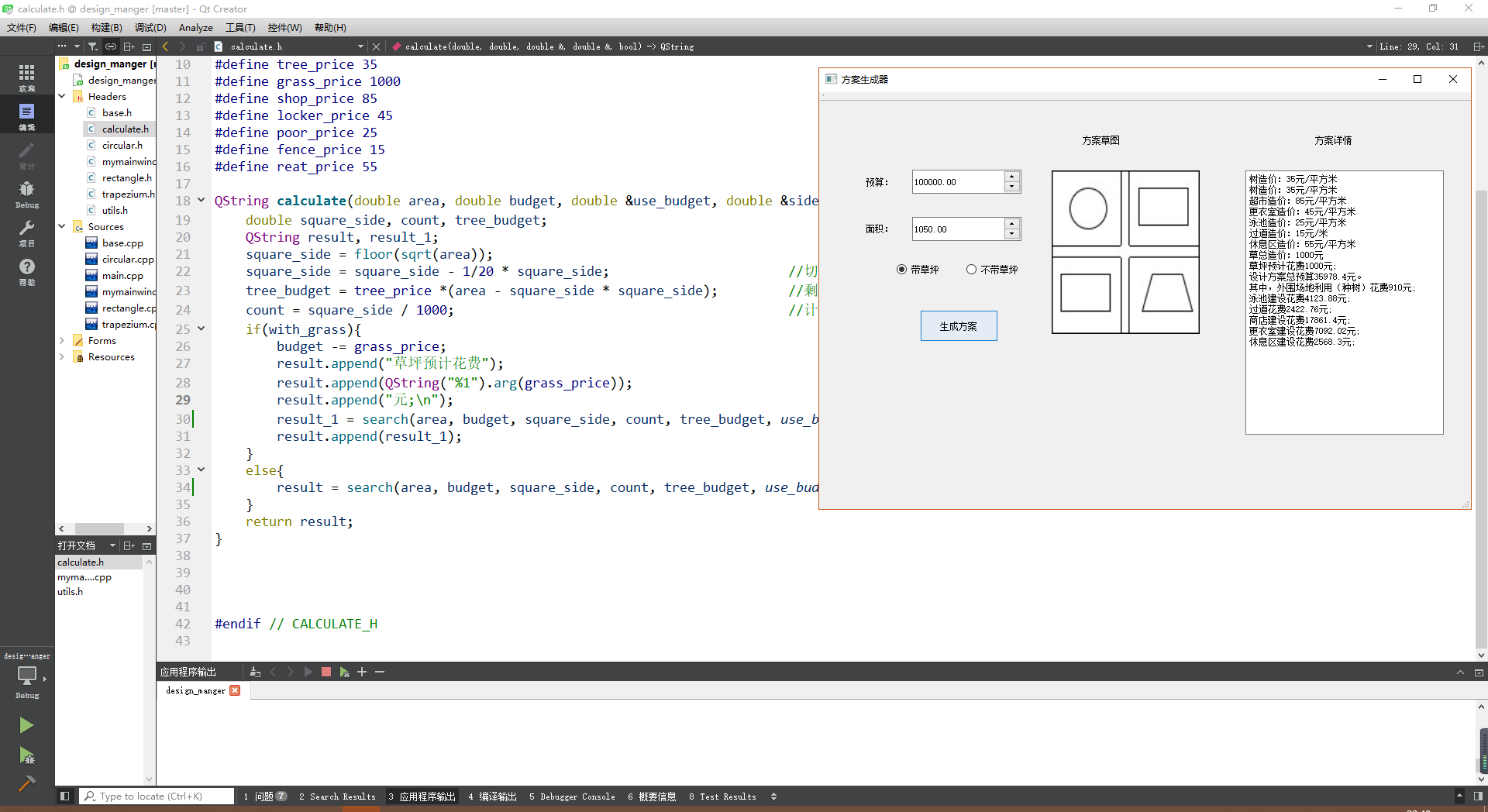
#endif // UTILS\_H

**测试数据：**

预算100000.00

面积1050.00

带草坪

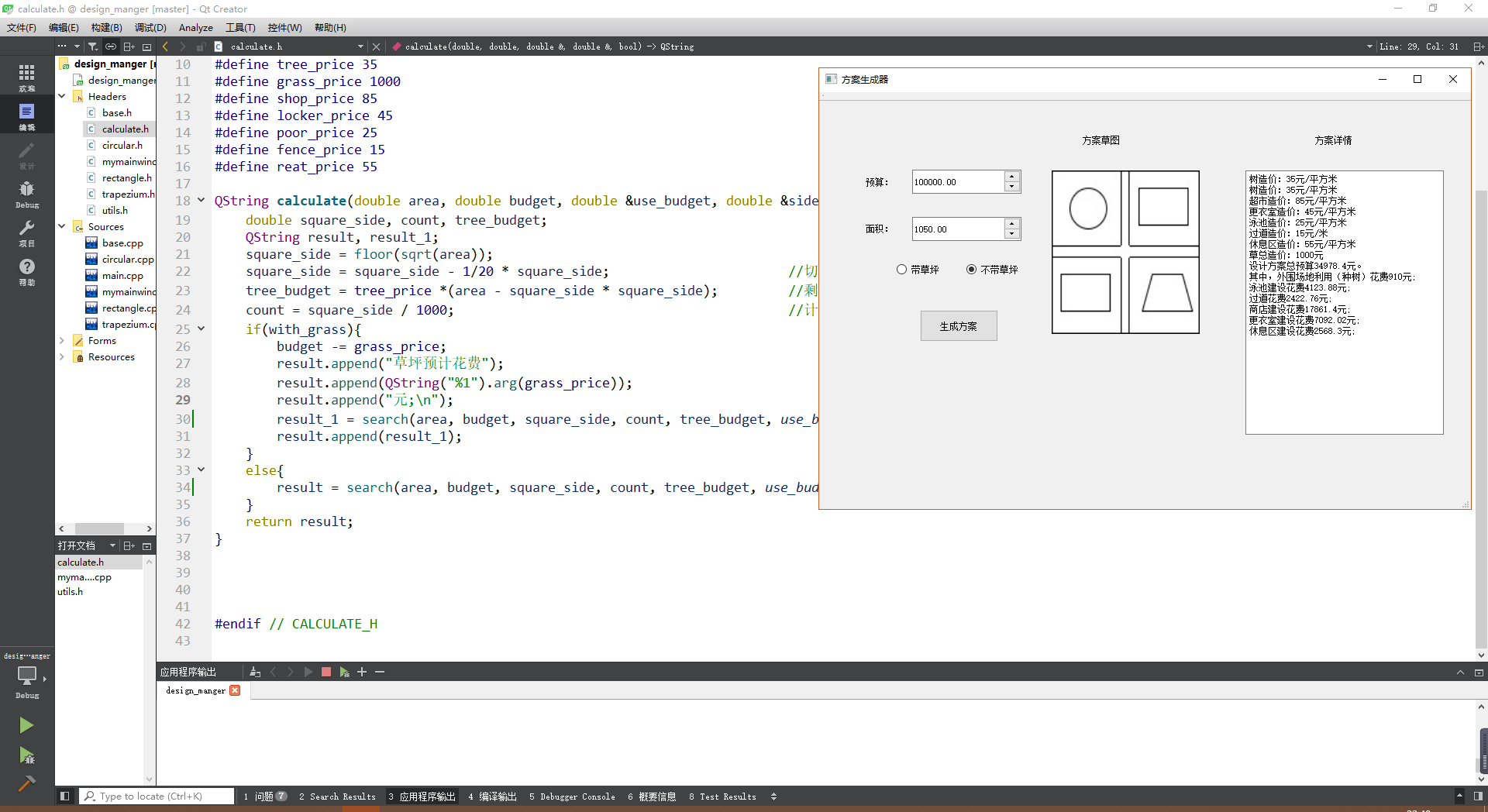


**测试数据：**

预算100000.00

面积1050.00

不带草坪

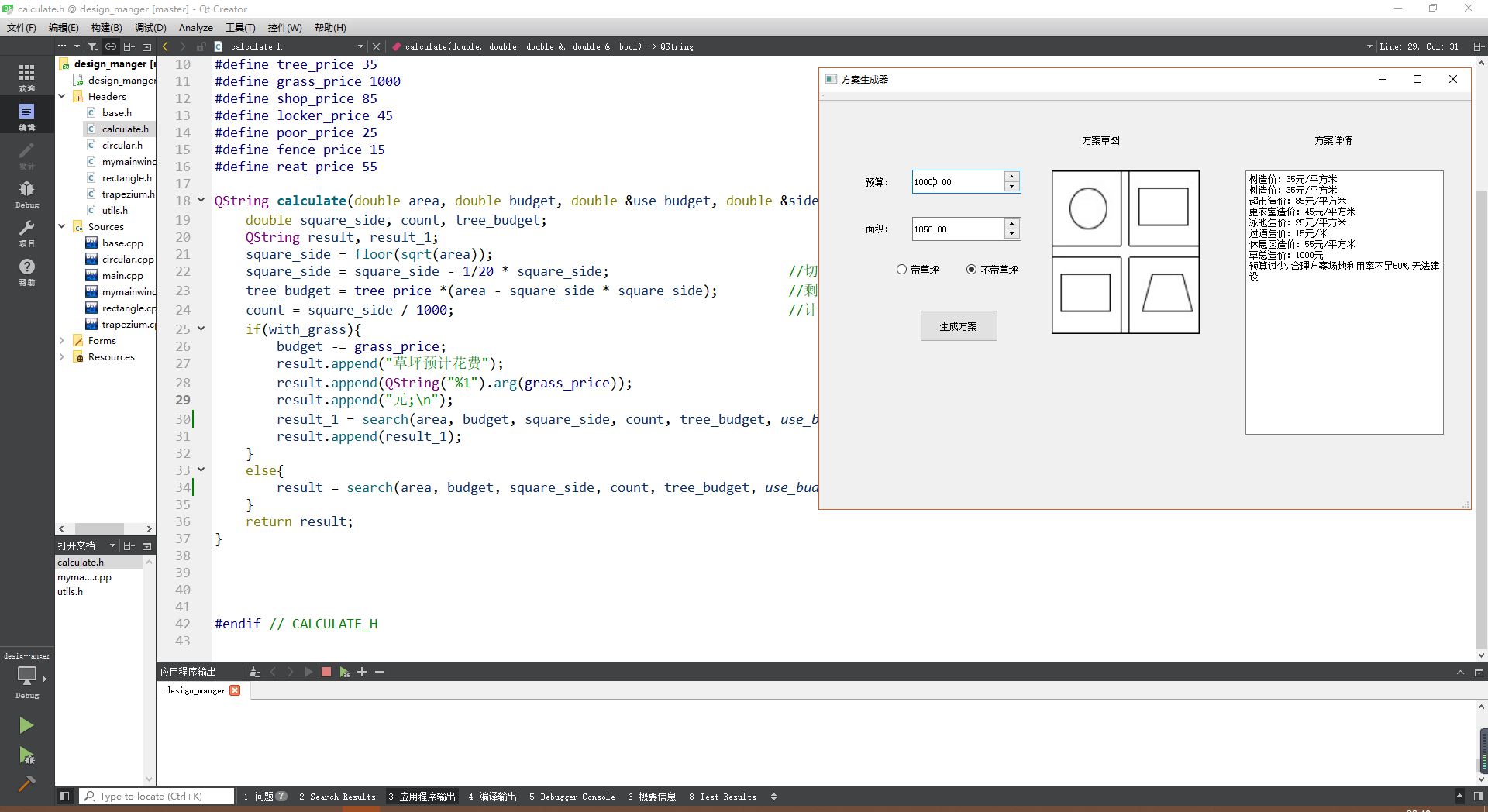


**测试数据：**

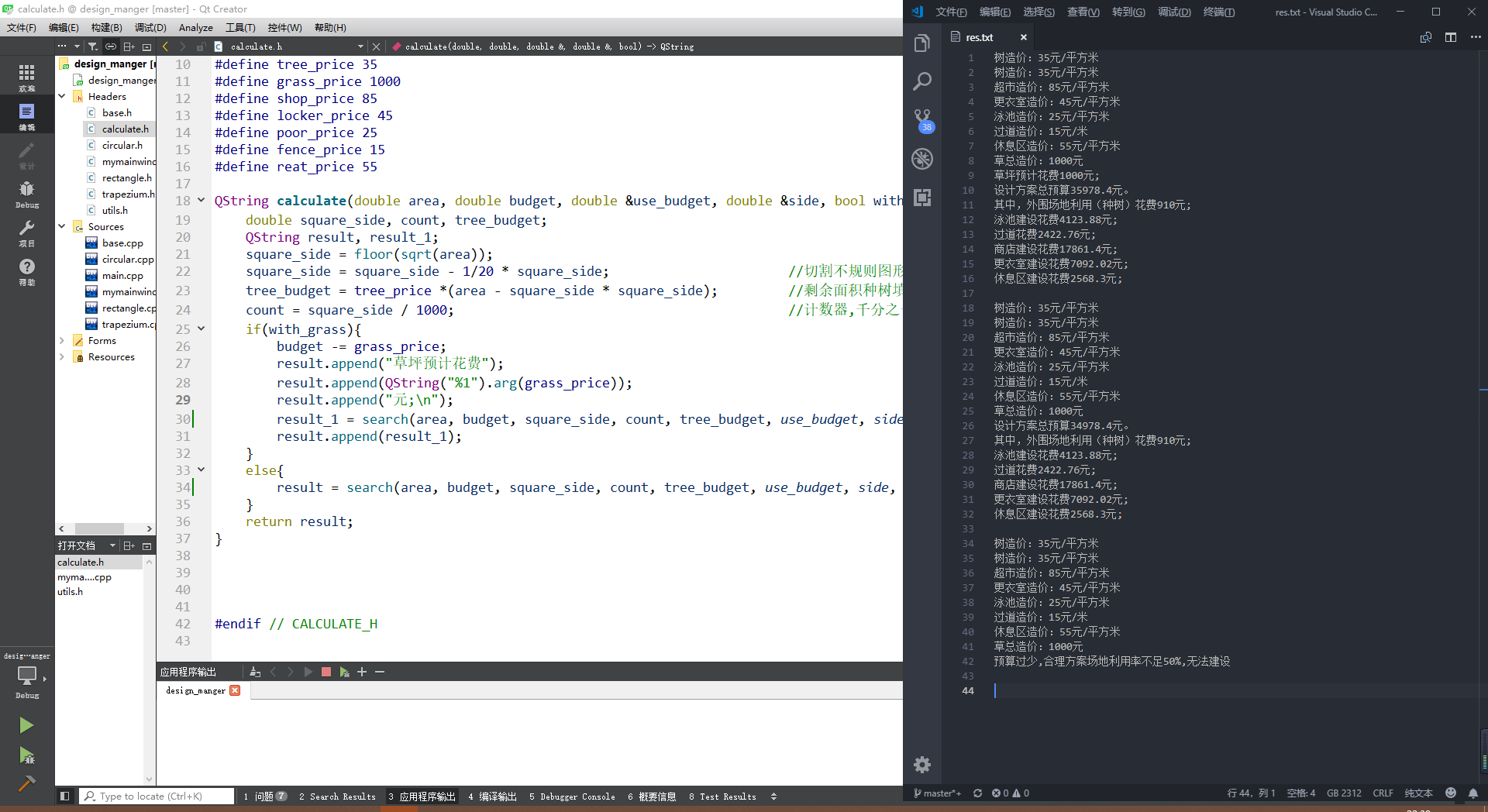
预算10000.00

面积1050.00

不带草坪



**文本输出：**



# 实验总结

通过本次实验，我们了解了软件项目是如何开展的，并学会了如何团队合作分工完成一个大项目。首先分析用户需求，提取有用信息并决定大方向。再学习并深入了解了如何编写界面、整合组件、实现逻辑的过程，熟练掌握继承派生、虚拟、多态等技术，利用QT库实现了友好的操作界面，以及输入输出流的实现，完成核心算法并将算法接入函数，完成项目的初步实现。最后进入测试环节，通过不断的换数据测试寻找整个项目代码的bug，再debug改代码，不断循环，最终完成整个项目。