

学号：2022141410193

姓名：王梟越

2 Makefile & Cmake

附上我的代码：

1. CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)

set(CMAKE_CXX_STANDARD 17)

project(HELLOWORLD LANGUAGES CXX)

add_executable(HELLOWORLD main.cpp stuinfo.h stuinfo.cpp)
```

2. stuinfo.h

```
#include <iostream>
#include <algorithm>
#include <cstring>

#ifndef STUINFO
#define STUINFO

struct stuinfo{
    char name[20];
    double score[3];
    double ave;
};

void inputstu(stuinfo stu[] , int n);
// asks the user to enter each of the preceding items of
//information to set the corresponding members of the structure.
void showstu(stuinfo stu[] , int n); //displays the contents of the structure, one
student one line.
void sortstu(stuinfo stu[] , int n); //sorts in descending order of average of three
scores.
bool findstu(stuinfo stu[] , int n, char ch[]); //finds if given characters is the
student's name.
#endif
```

3. stuinfo.cpp

```

#include <iostream>
#include <algorithm>
#include <cstring>
#include "stuinfo.h"

bool cmp(stuinfo a,stuinfo b){
    return a.ave>b.ave;
}

void inputstu(stuinfo stu[] , int n){
    for(int k=0;k<n;k++){
        double sum=0;
        std::cin>>stu[k].name;
        for(int i=0;i<3;i++)std::cin>>stu[k].score[i],sum+=stu[k].score[i];
        stu[k].ave=sum/3;
    }
}

void showstu(stuinfo stu[] , int n){
    for(int k=0;k<n;k++){
        std::cout<<"name:"<<stu[k].name<<" score:";
        for(int i=0;i<3;i++)std::cout<<stu[k].score[i]<<" ";
        std::cout<<"ave:"<<stu[k].ave<<std::endl;
    }
}

void sortstu(stuinfo stu[] , int n){
    std::sort(stu,stu+n,cmp);
}

bool findstu(stuinfo stu[] , int n, char ch[]){
    bool isfind=false;
    for(int k=0;k<n;k++){
        if(strcmp(stu[k].name,ch)==0){
            isfind=true;
            break;
        }
    }
    return isfind;
}

```

4. main.cpp

```

#include "stuinfo.h"
#include <iostream>
#include <algorithm>
#include <cstring>
const int MAXN=1e5+3;
int n;
char name[MAXN];
stuinfo stu[MAXN];

```

```

int main(){
    std::cout<<"Enter one integer n to represent the number of students\n";
    std::cin>>n;
    inputstu(stu,n);
    showstu(stu,n);
    sortstu(stu,n);
    showstu(stu,n);
    std::cout<<"Enter a name to check if the students include he\n";
    std::cin>>name;
    if(findstu(stu,n,name)){
        std::cout<<"Existed"<<std::endl;
    }else{
        std::cout<<"Not Existed"<<std::endl;
    }
    return 0;
}

```

附上过程图：

```

root@e199b59b582e:/ws/code/HELLOWORLD/build# cmake ../src
-- Configuring done
-- Generating done
-- Build files have been written to: /ws/code/HELLOWORLD/build
root@e199b59b582e:/ws/code/HELLOWORLD/build# cmake --build .
Scanning dependencies of target HELLOWORLD
[ 33%] Building CXX object CMakeFiles/HELLOWORLD.dir/main.cpp.o
[ 66%] Building CXX object CMakeFiles/HELLOWORLD.dir/stuinfo.cpp.o
[100%] Linking CXX executable HELLOWORLD
[100%] Built target HELLOWORLD
root@e199b59b582e:/ws/code/HELLOWORLD/build# ./HELLOWORLD
Enter one integer n to represent the number of students
3
a 1 2 3
b 2 3 4
c 3 4 5
name:a score:1 2 3 ave:2
name:b score:2 3 4 ave:3
name:c score:3 4 5 ave:4
name:c score:3 4 5 ave:4
name:b score:2 3 4 ave:3
name:a score:1 2 3 ave:2
Enter a name to check if the students include he
e
Not Existed
root@e199b59b582e:/ws/code/HELLOWORLD/build#

```

3 types

问答题

1. static 用法和作用
2. 什么是隐式转换, 如何消除隐式转换?

1. static 在C/Cpp中的共同特性有:

修饰全局变量或者函数, 使其只在该文件中可见。

修饰局部变量, 使其只在函数内部可见, 即使函数被多次调用, 变量也只会被初始化一次, 保存其值不变。

2. static 在CPP中的特性有:

附上我的代码:

```
#include <iostream>

class Point{
public:
    Point(int _x,int _y):x(_x),y(_y){
        count++;
    }
    Point(Point& p){
        x=p.x;
        y=p.y;
        count++;
    }
    int retx(){
        return x;
    }
    int rety(){
        return y;
    }
    //作用于类的函数成员
    static void printcnt(){
        std::cout<<count<<std::endl;
    }
private:
    int x,y;
    static int count;//作用于类的数据成员
};

int Point::count=0;//只能在类外初始化

int main(){
    Point A(114,514);
    Point::printcnt();//可用域操作符调用
    Point B(A);
    B.printcnt();//可用类名调用
    return 0;
}
```

附上运行结果：

```
1
2
root@e199b59b582e: /ws#
```

综上所述我们可以看出

- 静态数据成员由该类所有对象共同维护和使用，可实现同一个类的不同对象之间的数据共享。
- 静态成员函数由该类所有对象共同拥有，为所有对象共享

什么是隐式转换, 如何消除隐式转换?

用一段代码解释什么是隐式转换, 如下

```
#include <iostream>

class num{
public:
    num(int _x):val(_x){}
    bool iseq(num y){
        return val==y.val;
    }
private:
    int val;
};

int main(){
    num a(114);
    if(a.iseq(114)){
        std::cout<<"隐式转换成功"<<std::endl;
    }else{
        std::cout<<"隐式转换失败"<<std::endl;
    }
    return 0;
}
```

在上面的代码中, 调用 `a.iseq()` 时, 将 `(int)114` 转换成了类 `num(114)`, 运行结果如下

```
隐式转换成功
root@e199b59b582e: /ws#
```

消除方法就是在构造函数前面加上关键字 `explicit`, 效果如下:

```

(int)114
int main(){
    num a(114)
    if(a.iseq(114)){
        std::cout<<"隐式转换成功"<<std::endl;
    }
}

```

不存在从 "int" 转换到 "num" 的适当构造函数 C/C++(415)

[查看问题 \(Alt+F8\)](#) [快速修复... \(Ctrl+.\)](#)

程序解释题

- 代码1

```

#include <iostream>

using std::cout;
using std::endl;

int main() {
    int num1 = 1234567890;
    int num2 = 1234567890;
    int sum = num1 + num2;
    cout << "sum = " << sum << endl;

    float f1 = 1234567890.0f;
    float f2 = 1.0f;
    float fsum = f1 + f2;
    cout << "fsum = " << fsum << endl;
    cout << "(fsum == f1) is " << (fsum == f1) << endl;

    float f = 0.1f;
    float sum10x = f + f + f + f + f + f + f + f + f + f;
    float mul10x = f * 10;

    cout<<"sum10x = "<< sum10x << endl;
    cout<<"mul10x = "<< mul10x << endl;
    cout<<"(sum10x == 1) is "<< (sum10x == 1.0) << endl;
    cout<<"(mul10x == 1) is "<< (mul10x == 1.0) << endl;
    return 0;
}

```

```

sum = -1825831516
fsum = 1.23457e+09
(fsum == f1) is 1
sum10x = 1
mul10x = 1
(sum10x == 1) is 0
(mul10x == 1) is 1
root@e199b59b582e:/ws#

```

运行结果如上

1. `int` 的范围是32位, `num1+num2` 是超出了 `int` 的容纳范围导致爆负数
 2. `fsum` 的打印值是用的科学计数法, 当 `double` 类型的数据比较大时, 就会用科学计数法代表
 3. 判断 `fsum` 和 `f1` 是否相等时, 结果是相等, 因为在关于 `double` 类的数据的运算时, 会产生误差, 所以判定成了相等
 4. 打印都是1说明这两个浮点数在小数点后很高位才会有浮点误差
 5. 和1.0比较的结果不一样是因为累加的浮点误差大于做乘法的浮点误差, 而此时累加的误差足以使得与1.0比较结果为不相等
- 代码2

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

int main(){
    cout << fixed;
    float f1 = 1.0f;
    cout<<"f1 = "<<f1<<endl;

    float a = 0.1f;
    float f2 = a+a+a+a+a+a+a+a+a;
    cout<<"f2 = "<<f2<<endl;

    if(f1 == f2)
        cout << "f1 = f2" << endl;
    else
        cout << "f1 != f2" << endl;
    return 0;
}
```

```
f1 = 1.000000
f2 = 1.000000
f1 != f2
root@e199b59b582e: /ws#
```

运行结果如上。

导致判断 `f1` 和 `f2` 不相等的原因是 `f2` 在被赋值的过程中对 `a` 是进行了累加操作, 产生了较大的但是至多在小数点后6位后的浮点误差, 因此判断为不相等。

- 代码三

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

int main(){
```

```

int a, b;
double c, d;

a = 19.99 + 21.99;
b = (int)19.99 + (int)21.99;
c = 23 / 8;
d = 23 / 8.0;

cout << "a = " << a << endl;
cout << "b = " << b << endl;
cout << "c = " << c << endl;
cout << "d = " << d << endl;
cout << "0/0= " << 0/0 << endl;

return 0;
}

```

```

a = 41
b = 40
c = 2
d = 2.875
root@e199b59b582e:/ws#

```

```

17 cout << "0/0= " << 0/0 << endl;

```

出现异常。 ×
Arithmetic exception

运行结果如上。

1. `a` 在被赋值的时候，先是对浮点数19.99和21.99进行加法运算，然后将 `double` 类型转换成 `int` 类型赋值给 `a`。
2. `b` 在被赋值的时候，先是对浮点数19.99和21.99分别进行类型转换，变成 `int` 后在做加法赋值给 `b`，即将19和21做加法后的结果赋值给 `b`。
3. `c` 在被赋值的时候，先是对整数23和8进行整数除法，然后转换成 `double` 赋值给 `c`。
4. `d` 在被赋值的时候，先是对整数23进行类型转换，转换成 `double` 后和8.0做浮点数除法，最后将结果赋值给 `a`。
5. 计算 `0/0` 会产生 `arithmetic exception`，和数学一样0不能做分母。

4 Structs

对齐问题

```
#include <iostream>
// alignas 生效的情况
struct Info {
    uint8_t a;
    uint16_t b;
    uint8_t c;
};

struct alignas(4) Info2 {
    uint8_t a;
    uint16_t b;
    uint8_t c;
};

int main(){
    std::cout << sizeof(Info) << std::endl; // 6 2 + 2 + 2
    std::cout << alignof(Info) << std::endl; // 2
    std::cout << sizeof(Info2) << std::endl; // 8 4 + 4
    std::cout << alignof(Info2) << std::endl; // 4
    return 0;
}
```

生效情况中，`Info` 结构体是按照默认方式对齐的。故按照最大的16位/2字节进行对齐，每个数据占用2字节。

而 `Info2` 结构体使用了 `alignas()`，让其按照4字节对齐，这个过程并没有切割到原来的数据，并将原来的 `a` 和 `b` 化为第一层，将 `c` 化为第二层。

```
#include <iostream>
// alignas 生效的情况
struct Info {
    uint8_t a;
    uint32_t b;
    uint8_t c;
};

struct alignas(2) Info2 {
    uint8_t a;
    uint32_t b;
    uint8_t c;
};

int main(){
    std::cout << sizeof(Info) << std::endl; // 12 4 + 4 + 4
    std::cout << alignof(Info) << std::endl; // 4
    std::cout << sizeof(Info2) << std::endl; // 12 4 + 4 + 4
    std::cout << alignof(Info2) << std::endl; // 4
    return 0;
}
```

同理，没生效的情况中，

Info是默认按照最大的32位/4字节进行对齐的，对于三个数据就是占用了12字节。

Info2原本是按照4字节对齐的，当重新以2字节对齐时，会切割到数据b，那么此时alignas就没办法生效。

计算几何一则

附上代码

1 数学公式法+g++指令

```
//geo.h
#include <cmath>

#ifndef _GEO
#define _GEO

struct Point { double x, y; }; // 点
using Vec = Point; // 向量
struct Line { Point P; Vec v; }; // 直线（点向式）
struct Seg { Point A, B; }; // 线段（存两个端点）
struct Circle { Point O; double r; }; // 圆（存圆心和半径）
const Point O = {0, 0}; // 原点
const Line Ox = {0, {1, 0}}, Oy = {0, {0, 1}}; // 坐标轴
const double PI = acos(-1), EPS = 1e-9; //PI 与偏差值

std::pair<bool, Point> barycenter(const Point &A, const Point &B, const Point &C);
std::pair<bool, Point> circumcenter(const Point &A, const Point &B, const Point &C);
std::pair<bool, Point> incenter(const Point &A, const Point &B, const Point &C);
std::pair<bool, Point> orthocenter(const Point &A, const Point &B, const Point &C);

#endif
```

```
//geo.cpp
#include <iostream>
#include <utility>
#include "geo.h"

bool istri(const Point &A, const Point &B, const Point &C){
    double s=C.y*(B.x-A.x)+A.y*(C.x-B.x)+B.y*(A.x-C.x);
    return s>EPS||s<=-EPS;
}

std::pair<bool, Point> barycenter(const Point &A, const Point &B, const Point &C){
    double x,y;
    x=(A.x+B.x+C.x)/3;
    y=(A.y+B.y+C.y)/3;
    if(istri(A,B,C))
        return std::make_pair(istri(A,B,C),(Point){x,y});
    else
```

```

        return std::make_pair(istri(A,B,C),(Point){0,0});
    }

std::pair<bool, Point> circumcenter(const Point &A, const Point &B, const Point &C){
    if(!istri(A,B,C)) return std::make_pair(istri(A,B,C),(Point){0,0});
    double x,y;
    double A1=2*(B.x-A.x);
    double B1=2*(B.y-A.y);
    double C1=B.x*B.x+B.y*B.y-A.x*A.x-A.y*A.y;
    double A2=2*(C.x-B.x);
    double B2=2*(C.y-B.y);
    double C2=C.x*C.x+C.y*C.y-B.x*B.x-B.y*B.y;
    x=((C1*B2)-(C2*B1))/((A1*B2)-(A2*B1));
    y=((A1*C2)-(A2*C1))/((A1*B2)-(A2*B1));
    return std::make_pair(istri(A,B,C),(Point){x,y});
}

std::pair<bool, Point> incenter(const Point &A, const Point &B, const Point &C){
    if(!istri(A,B,C)) return std::make_pair(istri(A,B,C),(Point){0,0});
    double x,y,a,b,c;
    a=sqrt((B.x-C.x)*(B.x-C.x)+(B.y-C.y)*(B.y-C.y));
    b=sqrt((C.x-A.x)*(C.x-A.x)+(C.y-A.y)*(C.y-A.y));
    c=sqrt((A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y));
    x=(a*A.x+b*B.x+c*C.x)/(a+b+c);
    y=(a*A.y+b*B.y+c*C.y)/(a+b+c);
    return std::make_pair(istri(A,B,C),(Point){x,y});
}

std::pair<bool, Point> orthocenter(const Point &A, const Point &B, const Point &C){
    if(!istri(A,B,C)) return std::make_pair(istri(A,B,C),(Point){0,0});
    double x,y,a,b,c,cA,cB,cC;
    a=sqrt((B.x-C.x)*(B.x-C.x)+(B.y-C.y)*(B.y-C.y));
    b=sqrt((C.x-A.x)*(C.x-A.x)+(C.y-A.y)*(C.y-A.y));
    c=sqrt((A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y));
    cA=(b*b+c*c-a*a)/(2*b*c);
    cB=(c*c+a*a-b*b)/(2*c*a);
    cC=(a*a+b*b-c*c)/(2*a*b);
    if(cA&& cB&& cC){
        x=((a/cA)*A.x+(b/cB)*B.x+(c/cC)*C.x)/(a/cA+b/cB+c/cC);
        y=((a/cA)*A.y+(b/cB)*B.y+(c/cC)*C.y)/(a/cA+b/cB+c/cC);
    }else if(!cA){
        x=A.x;
        y=A.y;
    }else if(!cB){
        x=B.x;
        y=B.y;
    }else if(!cC){
        x=C.x;
        y=C.y;
    }

    return std::make_pair(istri(A,B,C),(Point){x,y});
}

```

```
}
```

```
#include <iostream>
#include <utility>
#include "geo.h"

int main(){
    double x,y;

    std::cout<<"enter two double integer as coordinates of A"<<std::endl;
    std::cin>>x>>y;
    Point A={x,y};

    std::cout<<"enter two double integer as coordinates of B"<<std::endl;
    std::cin>>x>>y;
    Point B={x,y};

    std::cout<<"enter two double integer as coordinates of C"<<std::endl;
    std::cin>>x>>y;
    Point C={x,y};

    if(barycenter(A,B,C).first)
        std::cout<<"barycenter: ("<<barycenter(A,B,C).second.x<<","
<<barycenter(A,B,C).second.y<<")"<<std::endl;
    else
        std::cout<<"Illegal triangular"<<std::endl;

    if(circumcenter(A,B,C).first)
        std::cout<<"circumcenter: ("<<circumcenter(A,B,C).second.x<<","
<<circumcenter(A,B,C).second.y<<")"<<std::endl;
    else
        std::cout<<"Illegal triangular"<<std::endl;

    if(incenter(A,B,C).first)
        std::cout<<"incenter: ("<<incenter(A,B,C).second.x<<","
<<incenter(A,B,C).second.y<<")"<<std::endl;
    else
        std::cout<<"Illegal triangular"<<std::endl;

    if(orthocenter(A,B,C).first)
        std::cout<<"orthocenter: ("<<orthocenter(A,B,C).second.x<<","
<<orthocenter(A,B,C).second.y<<")"<<std::endl;
    else
        std::cout<<"Illegal triangular"<<std::endl;

    return 0;
}
```

运行情况:

```

● root@e199b59b582e:/ws/code/Assignment_2# g++ -shared -fPIC -o libfunction.so geo.cpp
● root@e199b59b582e:/ws/code/Assignment_2# g++ -o main -L. main.cpp -lfunction
● root@e199b59b582e:/ws/code/Assignment_2# ./main
enter two double integer as coordinates of A
0 0
enter two double integer as coordinates of B
0 3
enter two double integer as coordinates of C
4 0
barycenter:(1.33333,1)
circumcenter:(2,1.5)
incenter:(1,1)
orthocenter:(0,0)

```

```

● root@e199b59b582e:/ws/code/Assignment_2# ./main
enter two double integer as coordinates of A
0 0
enter two double integer as coordinates of B
1 1
enter two double integer as coordinates of C
2 2
Illegal triangular
Illegal triangular
Illegal triangular
Illegal triangular

```

2 利用头文件中的一系列定义+CMAKE

geo.h

```

#include <cmath>

#ifndef _GEO
#define _GEO

struct Point { double x, y; }; // 点
using Vec = Point; // 向量
struct Line { Point P; Vec v; }; // 直线（点向式）
struct Seg { Point A, B; }; // 线段（存两个端点）
struct Circle { Point O; double r; }; // 圆（存圆心和半径）
const Point O = {0, 0}; // 原点
const Line Ox = {0, {1, 0}}, Oy = {0, {0, 1}}; // 坐标轴
const double PI = acos(-1), EPS = 1e-9; //PI 与偏差值

std::pair<bool, Point> barycenter(const Point &A, const Point &B, const Point &C);
std::pair<bool, Point> circumcenter(const Point &A, const Point &B, const Point &C);
std::pair<bool, Point> incenter(const Point &A, const Point &B, const Point &C);
std::pair<bool, Point> orthocenter(const Point &A, const Point &B, const Point &C);

#endif

```

geo.cpp

```

#include <iostream>
#include <utility>
#include "geo.h"

double dot_p(Vec a,Vec b){
    return a.x*b.x+a.y*b.y;
}

double crs_p_m(Vec a,Vec b){
    return a.x*b.y-b.x*a.y;
}

double module(Vec v){
    return sqrt(v.x*v.x+v.y*v.y);
}

std::pair<bool,Point> intersection(const Line& A,const Line& B){
    double s,t,x,y;
    Vec tmp={B.P.x-A.P.x,B.P.y-A.P.y};
    s=crs_p_m(A.v,B.v);
    if(s>=-EPS&&s<=EPS) return std::make_pair(false,(Point){0,0});
    t=crs_p_m(tmp,B.v);
    x=A.P.x+(t/s)*A.v.x;
    y=A.P.y+(t/s)*A.v.y;
    return std::make_pair(true,(Point){x,y});
}

bool istri(const Point &A, const Point &B, const Point &C){
    double s=C.y*(B.x-A.x)+A.y*(C.x-B.x)+B.y*(A.x-C.x);
    return s>EPS||s<=-EPS;
}

std::pair<bool, Point> barycenter(const Point &A, const Point &B, const Point &C){
    double x,y;
    x=(A.x+B.x+C.x)/3;
    y=(A.y+B.y+C.y)/3;
    if(istri(A,B,C))
        return std::make_pair(istri(A,B,C),(Point){x,y});
    else
        return std::make_pair(istri(A,B,C),(Point){0,0});
}

std::pair<bool, Point> circumcenter(const Point &A, const Point &B, const Point &C){
    if(!istri(A,B,C)) return std::make_pair(istri(A,B,C),(Point){0,0});

    Point D={(A.x+B.x)/2,(A.y+B.y)/2};
    Point E={(C.x+B.x)/2,(C.y+B.y)/2};
    Vec BA={A.x-B.x,A.y-B.y};
    Vec BC={C.x-B.x,C.y-B.y};
    Vec v1={-BA.y,BA.x};
    Vec v2={-BC.y,BC.x};
    Line l1={{D.x,D.y},v1};

```

```

    Line l2={{E.x,E.y},v2};
    auto ans=intersection(l1,l2);

    return std::make_pair(istri(A,B,C),ans.second);
}

std::pair<bool, Point> incenter(const Point &A, const Point &B, const Point &C){
    if(!istri(A,B,C)) return std::make_pair(istri(A,B,C),(Point){0,0});

    Vec BA={A.x-B.x,A.y-B.y};
    Vec BC={C.x-B.x,C.y-B.y};
    Vec CA={A.x-C.x,A.y-C.y};
    Vec v1={BA.x/module(BA)+BC.x/module(BC),BA.y/module(BA)+BC.y/module(BC)};
    Vec v2={CA.x/module(CA)-BC.x/module(BC),CA.y/module(CA)-BC.y/module(BC)};
    Line l1={{B.x,B.y},v1};
    Line l2={{C.x,C.y},v2};
    auto ans=intersection(l1,l2);

    return std::make_pair(istri(A,B,C),ans.second);
}

std::pair<bool, Point> orthocenter(const Point &A, const Point &B, const Point &C){
    if(!istri(A,B,C)) return std::make_pair(istri(A,B,C),(Point){0,0});

    Vec BA={A.x-B.x,A.y-B.y};
    Vec BC={C.x-B.x,C.y-B.y};
    Vec v1={-BA.y,BA.x};
    Vec v2={-BC.y,BC.x};
    Line l1={{C.x,C.y},v1};
    Line l2={{A.x,A.y},v2};
    auto ans=intersection(l1,l2);

    return std::make_pair(istri(A,B,C),ans.second);
}

```

main.cpp

```

#include <iostream>
#include <utility>
#include "geo.h"

int main(){
    double x,y;

    std::cout<<"enter two double integer as coordinates of A"<<std::endl;
    std::cin>>x>>y;
    Point A={x,y};

    std::cout<<"enter two double integer as coordinates of B"<<std::endl;
    std::cin>>x>>y;
    Point B={x,y};
}

```

```

std::cout<<"enter two double integer as coordinates of C"<<std::endl;
std::cin>>x>>y;
Point C={x,y};

if(barycenter(A,B,C).first)
    std::cout<<"barycenter: ("<<barycenter(A,B,C).second.x<<","
<<barycenter(A,B,C).second.y<<")"<<std::endl;
else
    std::cout<<"Illegal triangular"<<std::endl;

if(circumcenter(A,B,C).first)
    std::cout<<"circumcenter: ("<<circumcenter(A,B,C).second.x<<","
<<circumcenter(A,B,C).second.y<<")"<<std::endl;
else
    std::cout<<"Illegal triangular"<<std::endl;

if(incenter(A,B,C).first)
    std::cout<<"incenter: ("<<incenter(A,B,C).second.x<<","
<<incenter(A,B,C).second.y<<")"<<std::endl;
else
    std::cout<<"Illegal triangular"<<std::endl;

if(orthocenter(A,B,C).first)
    std::cout<<"orthocenter: ("<<orthocenter(A,B,C).second.x<<","
<<orthocenter(A,B,C).second.y<<")"<<std::endl;
else
    std::cout<<"Illegal triangular"<<std::endl;

return 0;
}

```

CMakeLists.txt

```

cmake_minimum_required(VERSION 2.83)

set(CMAKE_CXX_STANDARD 17)

project(ASSIGNMENT_2_GEO_NEW LANGUAGES CXX)

include_directories(/usr/include ./include)

link_directories(./include)

set(SRC geo.cpp)

add_library(geo SHARED ${SRC})

add_executable(main main.cpp)

target_link_libraries(main geo)

```


以上方法是通过三角形四点的定义进行向量和直线的运算得到的四点的位置，运行结果如下

```
● root@e199b59b582e:/ws/code/Assignment_2_geo_final# cmake .
-- The CXX compiler identification is GNU 11.2.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/local/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /ws/code/Assignment_2_geo_final
● root@e199b59b582e:/ws/code/Assignment_2_geo_final# make
Scanning dependencies of target geo
[ 25%] Building CXX object CMakeFiles/geo.dir/geo.cpp.o
[ 50%] Linking CXX shared library libgeo.so
[ 50%] Built target geo
Scanning dependencies of target main
[ 75%] Building CXX object CMakeFiles/main.dir/main.cpp.o
[100%] Linking CXX executable main
[100%] Built target main
● root@e199b59b582e:/ws/code/Assignment_2_geo_final# ./main
enter two double integer as coordinates of A
0 0
enter two double integer as coordinates of B
1 1
enter two double integer as coordinates of C
2 0
barycenter:(1,0.333333)
circumcenter:(1,0)
incenter:(1,0.414214)
orthocenter:(1,1)
○ root@e199b59b582e:/ws/code/Assignment_2_geo_final#
```

```
● root@e199b59b582e:/ws/code/Assignment_2_geo_final# ./main
enter two double integer as coordinates of A
0 0
enter two double integer as coordinates of B
1 1
enter two double integer as coordinates of C
2 2
Illegal triangular
Illegal triangular
Illegal triangular
Illegal triangular
```

5 C++动态内存申请

- C++的内存分区

```
class A {
```

```

int num;
}

static int a; //(1)
auto b=0; //(2)

int main(){
    char s[]="abc"; //(3)
    char *p="123456"; //(4)。

    p1= (char *)malloc(10); // (5)
    A *a = new A(); // (6)
}

```

1. (1) 是分配在全局/静态存储区中的
2. (2) 是分配在栈中的
3. (3) 是分配在栈中的，而右边的字符串常量是在常量存储区的
4. (4) 是分配在栈中的，而右边的字符串常量是在常量存储区的
5. (5) 是分配在栈中的，而malloc分配出的空间是在堆中的
6. (6) 是分配在栈中的，而new分配出的空间是在堆中的

- 问答题

1. new 和 malloc 的区别



是的，我知道 `new` 和 `malloc()` 之间的区别。它们都用于动态分配内存，但它们的实现和用法不同。



1. `new` 是 C++ 中的运算符，而 `malloc()` 是 C 标准库中的函数。
2. `new` 能够自动地调用类的构造函数来初始化对象，而 `malloc()` 只是分配一块原始的内存块。
3. `new` 返回指向新对象的指针，而 `malloc()` 返回指向分配的内存块的指针。这意味着，如果您使用 `new` 分配内存，则不需要将指针强制转换为特定的类型，因为它已经是正确的类型。
4. `new` 可以抛出异常，如果分配失败，会抛出 `std::bad_alloc` 异常。而 `malloc()` 只是返回一个空指针，不会抛出异常。
5. `delete` 运算符用于释放通过 `new` 分配的内存，而 `free()` 函数用于释放通过 `malloc()` 分配的内存。

总的来说，`new` 和 `malloc()` 的主要区别在于它们的实现和用法，以及 `new` 可以自动调用构造函数来初始化对象。在 C++ 中，建议使用 `new` 和 `delete` 来进行动态内存分配和释放，因为它们可以更好地与 C++ 的特性（如类和构造函数）结合使用，并且它们在错误处理方面更为强大。

1

综上，我们总结一下就是 `new` 充分体现了 C++ 面向对象的特点，以类来实现。

2. `delete p`、`delete[] p`、`allocator` 都有什么作用？

`delete p`、`delete[] p`、`allocator` 都有什么作用？



`delete p`、`delete[] p` 和 `allocator` 都是 C++ 中用于管理动态内存的机制。



1. `delete p`：用于释放由单个对象使用 `new` 分配的内存。这将调用指向 `p` 指针指向对象的析构函数，然后释放分配的内存。如果使用 `delete` 释放一个动态分配的数组，结果是未定义的行为。
2. `delete[] p`：用于释放使用 `new[]` 分配的动态数组的内存。它将调用指向数组中每个元素的析构函数，然后释放整个数组的内存。
3. `allocator`：用于在 C++ 中分配和释放内存。它是一个通用的内存分配器，它提供了可以使用的标准分配接口，例如 `allocate()` 和 `deallocate()`。

这些机制都是管理动态内存的关键，它们可以帮助你程序运行时分配和释放内存，以满足程序的内存需求。在使用它们时，需要注意避免内存泄漏和释放错误的内存块，以确保程序的正确性和健壮性。

2

其中具体讲一下 `allocate()` 和 `deallocate()` 的作用

在C++中, `allocate()` 函数是 `allocator` 类的一个成员函数, 用于分配内存块。

```
#include <iostream>
#include <memory>

int main() {
    std::allocator<int> alloc;
    int* ptr = alloc.allocate(10);

    for(int i = 0; i < 10; i++){
        *(ptr + i) = i;
    }

    for(int i = 0; i < 10; i++){
        std::cout << *(ptr + i) << " ";
    }

    alloc.deallocate(ptr, 10);
    return 0;
}
```

在上面的代码中, 我们可以看到用类 `allocator` 的两个成员函数 `allocate()` 和 `deallocate()` 可以进行内存的分配和删除。

3. `malloc` 申请的存储空间能用 `delete` 释放吗?

当然不行。 `new` 和 `malloc` 分配空间的方式都不一样, 自然 `delete` 和 `free` 的删除方式也不一样。

4. `malloc` 与 `free` 的实现原理?

再一次请教chatgpt



`malloc()` 和 `free()` 是 C 语言标准库提供的内存分配和释放函数，它们的实现原理可以简单地描述为下面的几个步骤：



1. 当调用 `malloc()` 函数时，系统会从堆中分配一块指定大小的内存空间，并返回该内存块的起始地址。堆是程序在运行时可以动态分配的内存空间。
2. 内存空间的分配通常会带有一定的开销，比如维护堆的状态信息，以及保证内存对齐等。因此，`malloc()` 函数通常会分配比请求的内存大小更多的内存，以便用于存储这些开销和其他状态信息。
3. 内存分配后，`malloc()` 函数会将内存块的起始地址返回给调用者。调用者可以使用该地址访问并操作内存块中的数据。
4. 当调用 `free()` 函数时，系统会将内存块标记为可用状态，并将其添加到堆的空闲链表中。在之后的 `malloc()` 调用中，系统可以重复使用该空间。
5. `free()` 函数通常不会释放整个内存块，而是将它标记为空闲状态，以便后续的 `malloc()` 调用可以使用该内存块。这样可以避免频繁的内存分配和释放操作，提高程序的性能。

需要注意的是，`malloc()` 和 `free()` 是 C 标准库提供的函数，在 C++ 中并不建议使用它们，而是应该使用 C++ 提供的 `new` 和 `delete` 运算符来进行内存分配和释放。`new` 和 `delete` 的内部实现方式不同于 `malloc()` 和 `free()`，它们可以自动调用对象的构造和析构函数，更加适合 C++ 中的面向对象编程。

3

`malloc()` 使用了内存分配相关的算法，整体的思想就是从系统中申请一块大小适合的内存自己管理，这块内存被称为**内存池**。

内存池相关的算法十分复杂，是还在发展的，还在被研究的话题。

- Exercise

RandomGenerator.h

```
#ifndef _RANDOMGENERATOR
#define _RANDOMGENERATOR

int rdgnrt(int n);

#endif
```

RandomGenerator.cpp

```

#include<iostream>
#include <random>
#include "RandomGenerator.h"
int rdgnrt(int n){
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<int> ds(0, n-1);
    return ds(gen);
}

```

main.cpp

```

#include <iostream>
#include <algorithm>
#include "RandomGenerator.h"

int n,_max=-0x7fffffff,_min=0x7fffffff;

int main(){
    std::cin>>n;
    int* p=new int[n];
    for(int i=0;i<n;i++){
        *(p+i)=rdgnrt(n);
    }
    for(int i=0;i<n;i++){
        _max=std::max(_max,*(p+i));
        _min=std::min(_min,*(p+i));
    }
    std::cout<<_max<<std::endl;
    std::cout<<_min<<std::endl;
    delete[] p;
    return 0;
}

```

运行结果

```

root@e199b59b582e:/ws/code/Assignment_2_mem/build# cmake ../src
-- Configuring done
-- Generating done
-- Build files have been written to: /ws/code/Assignment_2_mem/build
root@e199b59b582e:/ws/code/Assignment_2_mem/build# cmake --build .
[100%] Built target Assignment_2_mem
root@e199b59b582e:/ws/code/Assignment_2_mem/build# ./Assignment_2_mem
114
113
0
root@e199b59b582e:/ws/code/Assignment_2_mem/build# ./Assignment_2_mem
6
5
2
root@e199b59b582e:/ws/code/Assignment_2_mem/build# ./Assignment_2_mem
514
513
0
root@e199b59b582e:/ws/code/Assignment_2_mem/build#

```

6 DEBUG & RELEASE

- 如何判断动态申请越界（C方式）

debug x86

1. (1)(2)(3) 全部注释，观察运行结果

```

Microsoft Visual Studio 调试控制台
addr:00CFE1B8
00CFE1B4:fffffffd
00CFE1B5:fffffffd
00CFE1B6:fffffffd
00CFE1B7:fffffffd
00CFE1B8:31
00CFE1B9:32
00CFE1BA:33
00CFE1BB:34
00CFE1BC:35
00CFE1BD:36
00CFE1BE:37
00CFE1BF:38
00CFE1C0:39
00CFE1C1:00
00CFE1C2:fffffffd
00CFE1C3:fffffffd
00CFE1C4:fffffffd
00CFE1C5:fffffffd
00CFE1C6:41
00CFE1C7:42
D:\VS\dy_over\Debug\dy_over.exe (进程 6840) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .

```

2. (1) 放开，(2)(3) 注释，观察运行结果

```
Microsoft Visual Studio 调试控制台
addr:01128DD0
01128DCC:ffffffffd
01128DCD:ffffffffd
01128DCE:ffffffffd
01128DCF:ffffffffd
01128DD0:31
01128DD1:32
01128DD2:33
01128DD3:34
01128DD4:35
01128DD5:36
01128DD6:37
01128DD7:38
01128DD8:39
01128DD9:00
01128DDA:61
01128ddb:ffffffffd
01128DDC:ffffffffd
01128DDd:ffffffffd
01128DDE:41
01128DDF:42

D:\VS\dy_over\Debug\dy_over.exe (进程 38732)已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```

3. (1)(3) 放开, (2) 注释, 观察运行结果

```
D:\VS\dy_over\Debug\dy_over.exe
addr:008D8DD0
008D8DCC:ffffffffd
008D8DCD:ffffffffd
008D8DCE:ffffffffd
008D8DCF:ffffffffd
008D8DD0:31
008D8DD1:32
008D8DD2:33
008D8DD3:34
008D8DD4:35
008D8DD5:36
008D8DD6:37
008D8DD7:38
008D8DD8:39
008D8DD9:00
008D8DDA:61
008D8ddb:ffffffffd
008D8DDC:ffffffffd
008D8DDd:ffffffffd
008D8DDE:41
008D8DDF:42
-
```

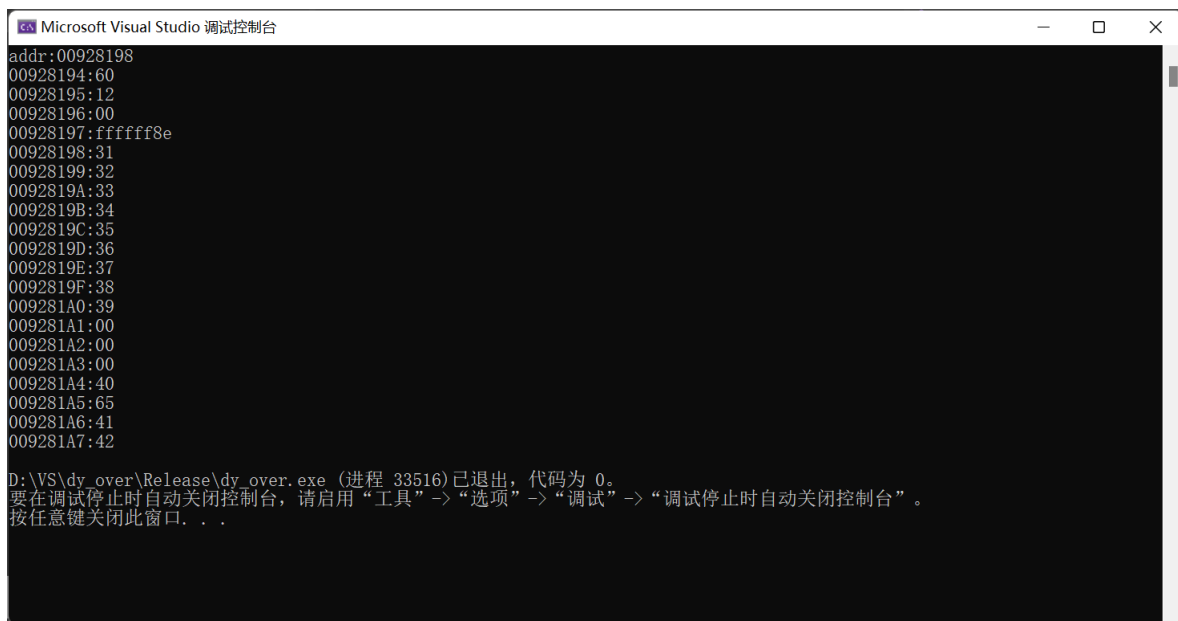
4. (1)(2)(3) 全部放开, 观察运行结果

```
Microsoft Visual Studio 调试控制台
addr:01667D28
01667D24:ffffffffd
01667D25:ffffffffd
01667D26:ffffffffd
01667D27:ffffffffd
01667D28:31
01667D29:32
01667D2A:33
01667D2B:34
01667D2C:35
01667D2D:36
01667D2E:37
01667D2F:38
01667D30:39
01667D31:00
01667D32:ffffffffd
01667D33:ffffffffd
01667D34:ffffffffd
01667D35:ffffffffd
01667D36:41
01667D37:42

D:\VS\dy_over\Debug\dy_over.exe (进程 7516)已退出, 代码为 0。
按任意键关闭此窗口。 . . .
```

release x86

1. (1)(2)(3) 全部注释，观察运行结果



```
Microsoft Visual Studio 调试控制台
addr:00928198
00928194:60
00928195:12
00928196:00
00928197:ffffff8e
00928198:31
00928199:32
0092819A:33
0092819B:34
0092819C:35
0092819D:36
0092819E:37
0092819F:38
009281A0:39
009281A1:00
009281A2:00
009281A3:00
009281A4:40
009281A5:65
009281A6:41
009281A7:42

D:\VS\dy_over\Release\dy_over.exe (进程 33516) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```

2. (1) 放开，(2)(3) 注释，观察运行结果



```
Microsoft Visual Studio 调试控制台
addr:00DDD120
00DDD11C:3b
00DDD11D:11
00DDD11E:00
00DDD11F:ffffff8e
00DDD120:31
00DDD121:32
00DDD122:33
00DDD123:34
00DDD124:35
00DDD125:36
00DDD126:37
00DDD127:38
00DDD128:39
00DDD129:00
00DDD12A:61
00DDD12B:5c
00DDD12C:61
00DDD12D:6e
00DDD12E:41
00DDD12F:42

D:\VS\dy_over\Release\dy_over.exe (进程 15392) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```

3. (1)(3) 放开，(2) 注释，观察运行结果

```
Microsoft Visual Studio 调试控制台

addr: 010CD2A8
010CD2A4:73
010CD2A5:00
010CD2A6:00
010CD2A7:ffffff8e
010CD2A8:31
010CD2A9:32
010CD2AA:33
010CD2AB:34
010CD2AC:35
010CD2AD:36
010CD2AE:37
010CD2AF:38
010CD2B0:39
010CD2B1:00
010CD2B2:61
010CD2B3:74
010CD2B4:61
010CD2B5:5c
010CD2B6:41
010CD2B7:42

D:\VS\dy_over\Release\dy_over.exe (进程 10588) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。...
```

4. (1)(2)(3) 全部放开，观察运行结果

```
Microsoft Visual Studio 调试控制台

addr: 007CD030
007CD02C:74
007CD02D:07
007CD02E:00
007CD02F:ffffff8e
007CD030:31
007CD031:32
007CD032:33
007CD033:34
007CD034:35
007CD035:36
007CD036:37
007CD037:38
007CD038:39
007CD039:00
007CD03A:fffffffd
007CD03B:77
007CD03C:65
007CD03D:72
007CD03E:41
007CD03F:42

D:\VS\dy_over\Release\dy_over.exe (进程 16312) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。...
```

Linux

1. (1)(2)(3) 全部注释，观察运行结果

```

● root@e199b59b582e:/ws/code/test/test_for_over# gcc -o 1 1.c
● root@e199b59b582e:/ws/code/test/test_for_over# ./1
addr:0x15df2a0
0x15df29c:00
0x15df29d:00
0x15df29e:00
0x15df29f:00
0x15df2a0:31
0x15df2a1:32
0x15df2a2:33
0x15df2a3:34
0x15df2a4:35
0x15df2a5:36
0x15df2a6:37
0x15df2a7:38
0x15df2a8:39
0x15df2a9:00
0x15df2aa:00
0x15df2ab:00
0x15df2ac:00
0x15df2ad:00
0x15df2ae:41
0x15df2af:42
○ root@e199b59b582e:/ws/code/test/test_for_over#

```

2. (1) 放开, (2)(3) 注释, 观察运行结果

```

● root@e199b59b582e:/ws/code/test/test_for_over# gcc -o 1 1.c
● root@e199b59b582e:/ws/code/test/test_for_over# ./1
addr:0xe452a0
0xe4529c:00
0xe4529d:00
0xe4529e:00
0xe4529f:00
0xe452a0:31
0xe452a1:32
0xe452a2:33
0xe452a3:34
0xe452a4:35
0xe452a5:36
0xe452a6:37
0xe452a7:38
0xe452a8:39
0xe452a9:00
0xe452aa:61
0xe452ab:00
0xe452ac:00
0xe452ad:00
0xe452ae:41
0xe452af:42
○ root@e199b59b582e:/ws/code/test/test_for_over#

```

3. (1)(3) 放开, (2) 注释, 观察运行结果

```

● root@e199b59b582e:/ws/code/test/test_for_over# gcc -o 1 1.c
● root@e199b59b582e:/ws/code/test/test_for_over# ./1
addr:0x22432a0
0x224329c:00
0x224329d:00
0x224329e:00
0x224329f:00
0x22432a0:31
0x22432a1:32
0x22432a2:33
0x22432a3:34
0x22432a4:35
0x22432a5:36
0x22432a6:37
0x22432a7:38
0x22432a8:39
0x22432a9:00
0x22432aa:61
0x22432ab:00
0x22432ac:00
0x22432ad:00
0x22432ae:41
0x22432af:42
○ root@e199b59b582e:/ws/code/test/test_for_over#

```

4. (1)(2)(3) 全部放开，观察运行结果

```

● root@e199b59b582e:/ws/code/test/test_for_over# gcc -o 1 1.c
● root@e199b59b582e:/ws/code/test/test_for_over# ./1
addr:0x17932a0
0x179329c:00
0x179329d:00
0x179329e:00
0x179329f:00
0x17932a0:31
0x17932a1:32
0x17932a2:33
0x17932a3:34
0x17932a4:35
0x17932a5:36
0x17932a6:37
0x17932a7:38
0x17932a8:39
0x17932a9:00
0x17932aa:fffffffd
0x17932ab:00
0x17932ac:00
0x17932ad:00
0x17932ae:41
0x17932af:42
○ root@e199b59b582e:/ws/code/test/test_for_over#

```

此外，在VS2022中的debug版本中以第三种方式越界时会弹出警告信息，而在Linux下用gdb调试也会产生警告。VS2022在申请动态内存前后设置了各4个字节的fffffff0，如果被修改就认为是越界。

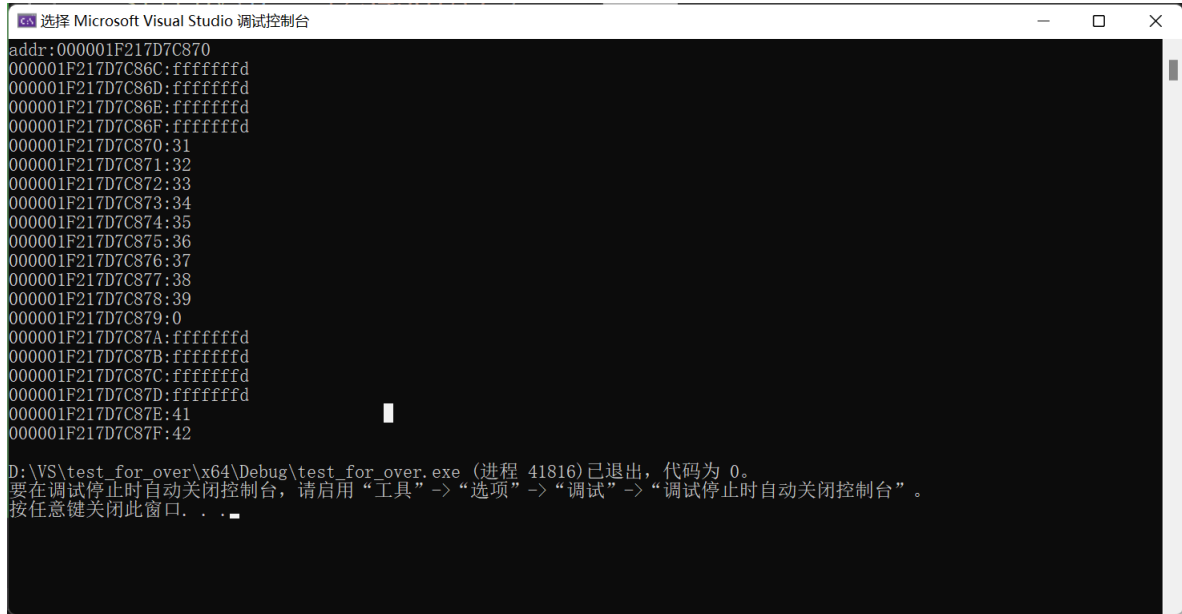
```
warning: Error disabling address space randomization: Operation not permitted
```

而在Release版本中就不会。

- 如何判断动态申请越界（C++方式）

Debug x86

1. (1)(2)(3) 全部注释，观察运行结果



```
选择 Microsoft Visual Studio 调试控制台
addr:000001F217D7C870
000001F217D7C86C:fffffffd
000001F217D7C86D:fffffffd
000001F217D7C86E:fffffffd
000001F217D7C86F:fffffffd
000001F217D7C870:31
000001F217D7C871:32
000001F217D7C872:33
000001F217D7C873:34
000001F217D7C874:35
000001F217D7C875:36
000001F217D7C876:37
000001F217D7C877:38
000001F217D7C878:39
000001F217D7C879:0
000001F217D7C87A:fffffffd
000001F217D7C87B:fffffffd
000001F217D7C87C:fffffffd
000001F217D7C87D:fffffffd
000001F217D7C87E:41
000001F217D7C87F:42

D:\VS\test_for_over\x64\Debug\test_for_over.exe (进程 41816) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```

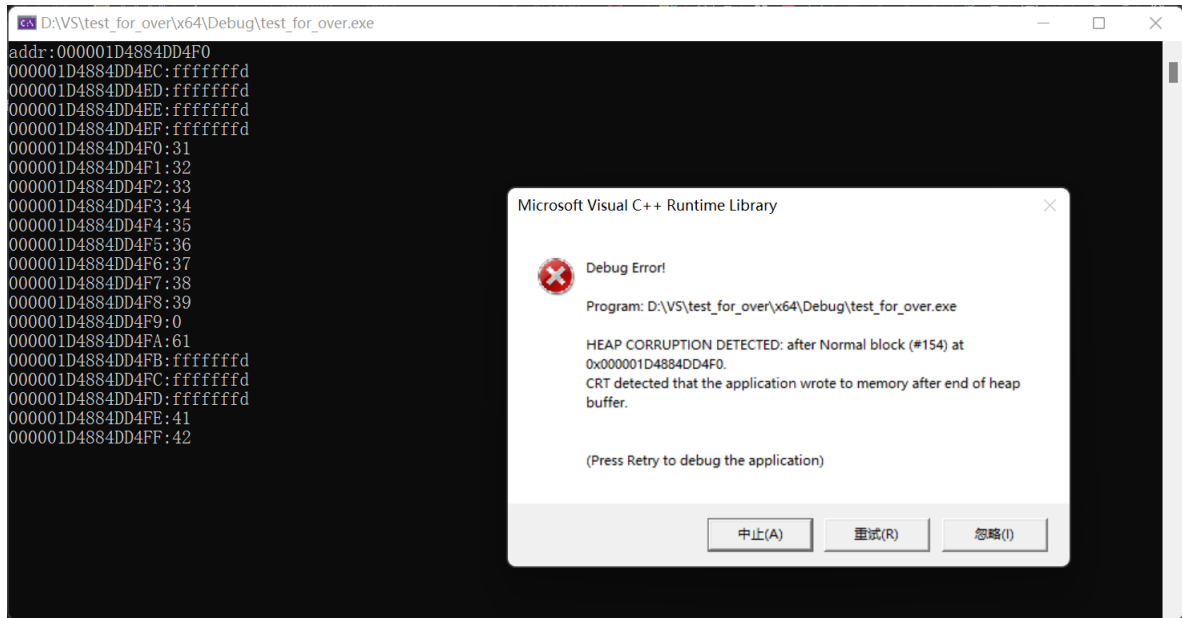
2. (1) 放开，(2)(3) 注释，观察运行结果



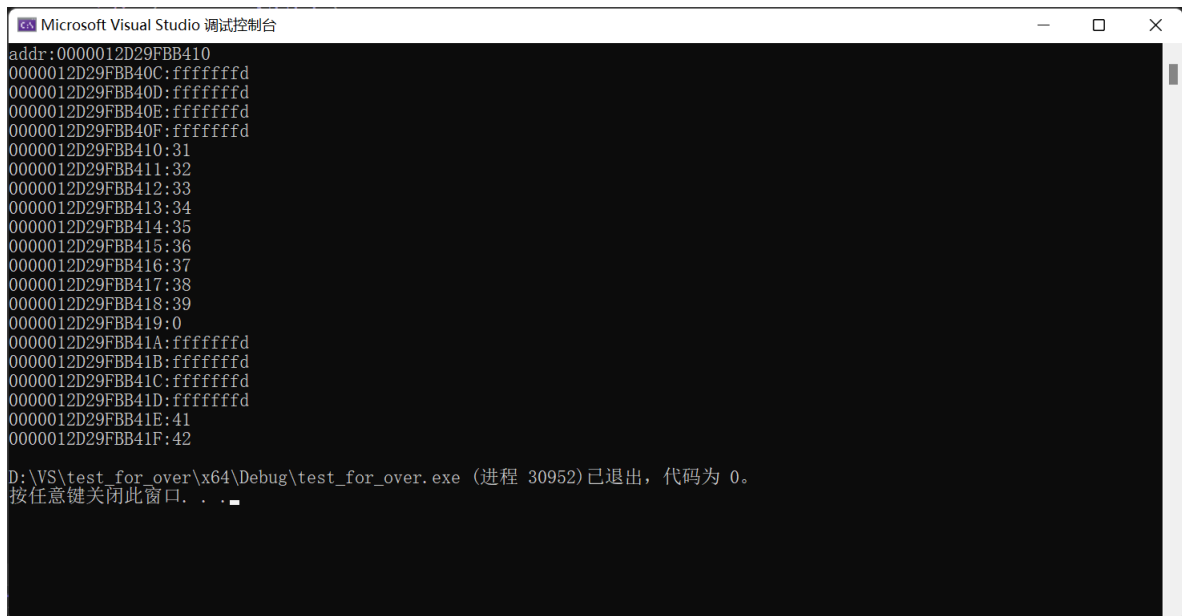
```
选择 Microsoft Visual Studio 调试控制台
addr:000001D245DFC7F0
000001D245DFC7EC:fffffffd
000001D245DFC7ED:fffffffd
000001D245DFC7EE:fffffffd
000001D245DFC7EF:fffffffd
000001D245DFC7F0:31
000001D245DFC7F1:32
000001D245DFC7F2:33
000001D245DFC7F3:34
000001D245DFC7F4:35
000001D245DFC7F5:36
000001D245DFC7F6:37
000001D245DFC7F7:38
000001D245DFC7F8:39
000001D245DFC7F9:0
000001D245DFC7FA:61
000001D245DFC7FB:fffffffd
000001D245DFC7FC:fffffffd
000001D245DFC7FD:fffffffd
000001D245DFC7FE:41
000001D245DFC7FF:42

D:\VS\test_for_over\x64\Debug\test_for_over.exe (进程 22548) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```

3. (1)(3) 放开, (2) 注释, 观察运行结果



4. (1)(2)(3) 全部放开, 观察运行结果



Release x86

1. (1)(2)(3) 全部注释，观察运行结果

```
Microsoft Visual Studio 调试控制台
addr:000001DD1A78320
000001DD1A7831C:5c
000001DD1A7831D:2b
000001DD1A7831E:0
000001DD1A7831F:ffffff96
000001DD1A78320:31
000001DD1A78321:32
000001DD1A78322:33
000001DD1A78323:34
000001DD1A78324:35
000001DD1A78325:36
000001DD1A78326:37
000001DD1A78327:38
000001DD1A78328:39
000001DD1A78329:0
000001DD1A7832A:5c
000001DD1A7832B:43
000001DD1A7832C:6f
000001DD1A7832D:6d
000001DD1A7832E:41
000001DD1A7832F:42

D:\VS\test_for_over\x64\Release\test_for_over.exe (进程 39844)已退出，代码为 0。
按任意键关闭此窗口。 . . .
```

2. (1) 放开，(2)(3) 注释，观察运行结果

```
选择 Microsoft Visual Studio 调试控制台
addr:0000022BD0954D50
0000022BD0954D4C:0
0000022BD0954D4D:30
0000022BD0954D4E:0
0000022BD0954D4F:ffffff96
0000022BD0954D50:31
0000022BD0954D51:32
0000022BD0954D52:33
0000022BD0954D53:34
0000022BD0954D54:35
0000022BD0954D55:36
0000022BD0954D56:37
0000022BD0954D57:38
0000022BD0954D58:39
0000022BD0954D59:0
0000022BD0954D5A:61
0000022BD0954D5B:0
0000022BD0954D5C:0
0000022BD0954D5D:0
0000022BD0954D5E:41
0000022BD0954D5F:42

D:\VS\test_for_over\x64\Release\test_for_over.exe (进程 4368)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```

3. (1)(3) 放开, (2) 注释, 观察运行结果

```
Microsoft Visual Studio 调试控制台
addr:000001F88F724870
000001F88F72486C:0
000001F88F72486D:9
000001F88F72486E:0
000001F88F72486F:ffffff96
000001F88F724870:31
000001F88F724871:32
000001F88F724872:33
000001F88F724873:34
000001F88F724874:35
000001F88F724875:36
000001F88F724876:37
000001F88F724877:38
000001F88F724878:39
000001F88F724879:0
000001F88F72487A:61
000001F88F72487B:0
000001F88F72487C:0
000001F88F72487D:0
000001F88F72487E:41
000001F88F72487F:42

D:\VS\test_for_over\x64\Release\test_for_over.exe (进程 37156)已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

4. (1)(2)(3) 全部放开, 观察运行结果

```
Microsoft Visual Studio 调试控制台
addr:0000019340664ED0
0000019340664ECC:0
0000019340664ECD:3c
0000019340664ECE:0
0000019340664ECF:ffffff96
0000019340664ED0:31
0000019340664ED1:32
0000019340664ED2:33
0000019340664ED3:34
0000019340664ED4:35
0000019340664ED5:36
0000019340664ED6:37
0000019340664ED7:38
0000019340664ED8:39
0000019340664ED9:0
0000019340664EDA:fffffffd
0000019340664EDB:0
0000019340664EDC:0
0000019340664EDD:0
0000019340664EDE:41
0000019340664EDF:42

D:\VS\test_for_over\x64\Release\test_for_over.exe (进程 14772)已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

Linux

1. (1)(2)(3) 全部注释，观察运行结果

```
root@e199b59b582e:/ws/code/test/test_for_over# g++ -o _1 1.cpp
root@e199b59b582e:/ws/code/test/test_for_over# ./_1
addr:0x1d1eeb0
0x1d1eeac:0
0x1d1eead:0
0x1d1eeae:0
0x1d1eeaf:0
0x1d1eeb0:31
0x1d1eeb1:32
0x1d1eeb2:33
0x1d1eeb3:34
0x1d1eeb4:35
0x1d1eeb5:36
0x1d1eeb6:37
0x1d1eeb7:38
0x1d1eeb8:39
0x1d1eeb9:0
0x1d1eeba:0
0x1d1eebb:0
0x1d1eebc:0
0x1d1eebd:0
0x1d1eebe:41
0x1d1eebf:42
root@e199b59b582e:/ws/code/test/test_for_over#
```

滚动鼠标轴或单击，开始截长图

滚动鼠标轴或单击，开始截长图

2. (1) 放开，(2)(3) 注释，观察运行结果

```
root@e199b59b582e:/ws/code/test/test_for_over# g++ -o _1 1.cpp
root@e199b59b582e:/ws/code/test/test_for_over# ./_1
addr:0x21d6eb0
0x21d6eac:0
0x21d6ead:0
0x21d6eae:0
0x21d6eaf:0
0x21d6eb0:31
0x21d6eb1:32
0x21d6eb2:33
0x21d6eb3:34
0x21d6eb4:35
0x21d6eb5:36
0x21d6eb6:37
0x21d6eb7:38
0x21d6eb8:39
0x21d6eb9:0
0x21d6eba:61
0x21d6ebb:0
0x21d6ebc:0
0x21d6ebd:0
0x21d6ebe:41
0x21d6ebf:42
root@e199b59b582e:/ws/code/test/test_for_over#
```

滚动鼠标轴或单击，开始截长图

3. (1)(3) 放开, (2) 注释, 观察运行结果

```
● root@e199b59b582e:/ws/code/test/test_for_over# g++ -o _1 1.cpp
● root@e199b59b582e:/ws/code/test/test_for_over# ./_1
addr:0x12e9eb0
0x12e9eac:0
0x12e9ead:0
0x12e9eae:0
0x12e9eaf:0
0x12e9eb0:31
0x12e9eb1:32
0x12e9eb2:33
0x12e9eb3:34
0x12e9eb4:35
0x12e9eb5:36
0x12e9eb6:37
0x12e9eb7:38
0x12e9eb8:39
0x12e9eb9:0
0x12e9eba:61
0x12e9ebb:0
0x12e9ebc:0
0x12e9ebd:0
0x12e9ebe:41
0x12e9ebf:42
○ root@e199b59b582e:/ws/code/test/test_for_over#
```

滚动鼠标轴或单击, 开始截长图

滚动鼠标轴或单击, 开始截长图

4. (1)(2)(3) 全部放开, 观察运行结果

```
● root@e199b59b582e:/ws/code/test/test_for_over# g++ -o _1 1.cpp
● root@e199b59b582e:/ws/code/test/test_for_over# ./_1
addr:0x1a8beb0
0x1a8beac:0
0x1a8bead:0
0x1a8beae:0
0x1a8beaf:0
0x1a8beb0:31
0x1a8beb1:32
0x1a8beb2:33
0x1a8beb3:34
0x1a8beb4:35
0x1a8beb5:36
0x1a8beb6:37
0x1a8beb7:38
0x1a8beb8:39
0x1a8beb9:0
0x1a8beba:fffffffd
0x1a8bebb:0
0x1a8bebc:0
0x1a8bebd:0
0x1a8bebe:41
0x1a8bebf:42
○ root@e199b59b582e:/ws/code/test/test_for_over#
```

滚动鼠标轴或单击, 开始截长图

滚动鼠标轴或单击, 开始截长图

warning: Error disabling address space randomization: Operation not permitted

在gdb下调试依然会有报错, 如上

结论与C风格相似，不再赘述

- 如何判断普通数组的越界访问 (C++)

Linux

代码如下

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

int a[10];
char b[10];

int main(){
    for(int i=0;i<10;i++){
        a[i]=i+1;
        b[i]=i+'A';
    }
    a[10]=114;
    a[15]=514;
    b[10]='h';
    b[13]='o';
    b[14]='m';
    b[15]='o';
    cout<<"addr(a):"<<hex<<(void*)a<<endl;
    for(int i=-4;i<16;i++){
        cout<<hex<<(void*)(a+i)<<":";
        cout<<dec<<int(a[i])<<endl;
    }
    cout<<"addr(b):"<<hex<<(void*)b<<endl;
    for(int i=-4;i<16;i++){
        cout<<hex<<(void*)(b+i)<<":";
        cout<<dec<<int(b[i])<<endl;
    }
    return 0;
}
```

```

● root@e199b59b582e:/ws/code/test/test_for_over# g++ -o 2 2.cpp
● root@e199b59b582e:/ws/code/test/test_for_over# ./2
addr(a):0x4041a0
0x404190:0
0x404194:0
0x404198:0
0x40419c:0
0x4041a0:1
0x4041a4:2
0x4041a8:3
0x4041ac:4
0x4041b0:5
0x4041b4:6
0x4041b8:7
0x4041bc:8
0x4041c0:9
0x4041c4:10
0x4041c8:114
0x4041cc:12126305
0x4041d0:6834761
0x4041d4:1869442816
0x4041d8:0
0x4041dc:514
addr(b):0x4041c8
0x4041c4:10
0x4041c5:0
0x4041c6:0
0x4041c7:0
0x4041c8:114
0x4041c9:0
0x4041ca:0
0x4041cb:0
0x4041cc:69
0x4041cd:70
0x4041ce:71
0x4041cf:72
0x4041d0:73
0x4041d1:74
0x4041d2:104
0x4041d3:0
0x4041d4:0
0x4041d5:111
0x4041d6:109
0x4041d7:111
○ root@e199b59b582e:/ws/code/test/test_for_over#

```

我们可以看到，`int` 数组的越界影响到了 `char` 数组的前几项。

Debug

```
Microsoft Visual Studio 调试控制台

addr(a):0099C138
0099C128:0
0099C12C:0
0099C130:0
0099C134:0
0099C138:1
0099C13C:2
0099C140:3
0099C144:4
0099C148:5
0099C14C:6
0099C150:7
0099C154:8
0099C158:9
0099C15C:10
0099C160:114
0099C164:1212630597
0099C168:6834761
0099C16C:1869442816
0099C170:0
0099C174:514
addr(b):0099C160
0099C15C:10
0099C15D:0
0099C15E:0
0099C15F:0
0099C160:114
0099C161:0
0099C162:0
0099C163:0
0099C164:69
0099C165:70
0099C166:71
0099C167:72
0099C168:73
0099C169:74
0099C16A:104
0099C16B:0
0099C16C:0
0099C16D:111
0099C16E:109
0099C16F:111

D:\VS\test_for_over2\Debug\test_for_over2.exe (进程 29156) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...
```

Release

```
Microsoft Visual Studio 调试控制台
addr(a):003E5388
003E5378:2
003E537C:1145258561
003E5380:1212630597
003E5384:6834761
003E5388:1869442817
003E538C:2
003E5390:3
003E5394:4
003E5398:5
003E539C:6
003E53A0:7
003E53A4:8
003E53A8:9
003E53AC:10
003E53B0:114
003E53B4:0
003E53B8:0
003E53BC:0
003E53C0:0
003E53C4:514
addr(b):003E537C
003E5378:2
003E5379:0
003E537A:0
003E537B:0
003E537C:65
003E537D:66
003E537E:67
003E537F:68
003E5380:69
003E5381:70
003E5382:71
003E5383:72
003E5384:73
003E5385:74
003E5386:104
003E5387:0
003E5388:1
003E5389:111
003E538A:109
003E538B:111

D:\VS\test_for_over2\Release\test_for_over2.exe (进程 13900)已退出，代码为 0。
按任意键关闭此窗口。 . . .
```

可看到在VS2022是和Linux环境一个结果。

7 计算器的实现

代码如下（单*.cpp文件实现）

```
#include <iostream>
#include <stack>
#include <queue>
#include <string>
#include <map>
#include <utility>
#include <cmath>
#include <random>
using namespace std;

stack<char> opt;
queue<string> out;
stack<double> num;
map<string,string> f;

bool isok;
```

```

int op;

bool isille(string x){
    for(int i=0;i<x.size();i++){
        if(!isalpha(x[i])){
            return false;
        }
    }
    return true;
}

void Postfix_Expression(string x){
    isok=1;op=0;
    //预处理
    string _x,t;
    for(int i=0;i<x.size();i++){
        if(!isalpha(x[i])){
            _x+=x[i];
        }else{
            t+=x[i];
            if(!isalpha(x[i+1]))
                if(t=="sqrt")
                    _x+=t,op=1,t="";
                else if(t=="exp")
                    _x+=t,op=2,t="";
                else if(t=="abs")
                    _x+=t,op=3,t="";
                else if(t=="max")
                    _x+=t,op=4,t="";
                else if(t=="random")
                    _x+=t,op=5,t="";
                else if(t=="sin")
                    _x+=t,op=6,t="";
                else if(f.find(t)!=f.end()){
                    _x+=f[t];t="";
                }else{
                    isok=0;
                    cout<<"The Variable does not exist!(error message)"<<endl;
                    t="";
                    return;
                }
        }
    }

    x=_x;

    while(!opt.empty())opt.pop();
    while(!out.empty())out.pop();
    string tmp;
    for(int i=0;i<x.size();i++){
        if(x[i]=='('){
            opt.push(x[i]);

```

```

}else if(x[i]==')'){
    if(opt.empty()){
        isok=0;
        cout<<"The bracket is not matching!(error message)"<<endl;
        return;
    }
    char t=opt.top();
    while(t!='('){
        opt.pop();
        string _t;
        _t+=t;
        out.push(_t);
        if(opt.empty()){
            isok=0;
            cout<<"The bracket is not matching!(error message)"<<endl;
            return;
        }
        t=opt.top();
    }
    if(t=='('){
        opt.pop();
    }
}else if(x[i]=='+' || (x[i]=='-'&&is0&&isdigit(x[i-1])) || x[i]=='*' || x[i]=='/' || x[i]=='^'){
    if(x[i]=='+' || (x[i]=='-'&&isdigit(x[i-1]))){
        while(!opt.empty()){
            char t=opt.top();
            if(t=='(')break;
            string _t;
            _t+=t;
            out.push(_t);
            opt.pop();
        }
        opt.push(x[i]);
    }else if(x[i]=='*' || x[i]=='/'){
        while(!opt.empty()){
            char t=opt.top();
            if(t=='+' || t=='-' || opt.empty() || t=='(')break;
            string _t;
            _t+=t;
            out.push(_t);
            opt.pop();
        }
        opt.push(x[i]);
    }else if(x[i]=='^'){
        opt.push('^');
    }
}else if(isdigit(x[i]) || x[i]=='.' || x[i]=='-'){
    tmp+=x[i];
    if(!(isdigit(x[i+1]) || x[i+1]=='.')){
        out.push(tmp);
        tmp="";
    }
}

```



```

    }
}
}
while(!opt.empty()){
    char t=opt.top();
    if(t=='('){
        cout<<"The bracket is not matching!(error message)"<<endl;
        isok=0;
        break;
    }
    string _t;
    _t+=t;
    out.push(_t);
    opt.pop();
}
}

double calc(){
    while(!num.empty())num.pop();
    while(!out.empty()){
        string tmp=out.front();
        out.pop();
        if(isdigit(tmp[0])||(tmp[0]=='-'&&tmp.size()>1)){
            double t=stod(tmp);
            num.push(t);
        }else{
            double t1=num.top();
            num.pop();
            double t2=num.top();
            num.pop();
            if(tmp==""){
                num.push(t1+t2);
            }else if(tmp=="-"){
                num.push(t2-t1);
            }else if(tmp=="*"){
                num.push(t1*t2);
            }else if(tmp=="/"){
                num.push(t2/t1);
            }else if(tmp=="^"){
                num.push(pow(t2,t1));
            }
        }
    }
}

if(op==0)
    return num.top();
else if(op==1)
    return sqrt(num.top());
else if(op==2)
    return exp(num.top());
else if(op==3)
    return abs(num.top());
else if(op==4){

```

```

        double t1=num.top();
        num.pop();
        double t2=num.top();
        num.pop();
        return max(t1,t2);
    }else if(op==5){
        random_device rd;
        mt19937 gen(rd());
        uniform_int_distribution<int> dis(1, 100);
        int random_number = dis(gen);
        return random_number;
    }else if(op==6){
        return sin(num.top());
    }else{
        return num.top();
    }
}

pair<bool,pair<string,string>> val(string x){
    int pos=x.find('=');
    if(pos==-1)return make_pair(false,make_pair("", "-1"));
    return make_pair(true,make_pair(x.substr(0,pos),x.substr(pos+1)));
}

int main(){

    f["PI"]=to_string(acos(-1));
    f["E"]=to_string(exp(1));

    string x;
    while(getline(cin,x)){
        if(!val(x).first){
            Postfix_Expression(x);
            if(isok)
                cout<<calc()<<endl;
        }else{
            if(val(x).second.first=="PI"||val(x).second.first=="E"){
                cout<<"This is constant value!(error message)"<<endl;
            }else if(isille(val(x).second.first))
                f[val(x).second.first]=val(x).second.second;
            else
                cout<<"The variable name is not valid!(error message)"<<endl;
        }
    }
    return 0;
}

```

下面简单描述一下代码的思路。

首先，写一个函数 `val` 来判断我们输入的表达式是不是赋值表达式，如果是，建立变量到值的映射，我用的是 STL 中的 `map` 实现，并且用 `isille` 判断变量名字是否合法；如果不是，则进入表达式的计算过程。

计算过程分为两个过程，第一个是处理中缀表达式为后缀表达式。在我们的函数 `Postfix_Expression()` 中，我们首先将表达式进行预处理，也就是将变量全部变成数字，若有调用数学函数，则记录下选择的是哪个函数。然后我们用一个栈 `opt` 和一个队列 `out` 来处理我们的表达式，思路是

1. 如果当前字符是数字，直接将它输出到队列 `out` 中。
2. 如果当前字符是左括号，将它压入栈 `opt` 中。
3. 如果当前字符是右括号，将栈 `opt` 中的元素依次弹出并输出到队列 `out` 中，直到遇到左括号。
4. 如果当前字符是运算符，我们需要进行以下处理：
 1. 如果当前运算符的优先级大于栈 `opt` 的栈顶的运算符，将它压入栈 `opt` 中。
 2. 如果当前运算符的优先级小于或等于栈 `opt` 的栈顶的运算符，将栈 `opt` 的栈顶的元素依次弹出并输出到队列 `out` 中，直到遇到一个优先级小于当前运算符的运算符或者栈为空，然后将当前运算符压入栈 `opt` 中。
 3. 如果当前运算符是指数运算符 (^)，由于它的优先级最高，所以它总是压入栈 `opt` 中，而不是弹出其他运算符。
5. 如果扫描完成后，栈 `opt` 中还有元素，将它们依次弹出并输出到队列 `out` 中。

以上的算法是处理后缀表达式的核心思路。

然后我们写 `calc` 函数，实现的功能是根据输出队列算出后缀表达式的值，以及根据调用数学函数的标记，算出最后的值。

值得说明的是，我还对负数进行了特殊处理，例如我的代码在计算 $3 - -3$ 的时候会识别成 3 减掉 -3 得到 6。

实现思想是在处理负号和减号的时候，假如 - 的左边是数字，则认为它是减号，否则算成负号单独处理。

1. 来自 [chatgpt](#)

2. 来自 [chatgpt](#)

3. 来自 [chatgpt](#)