Student Name: Gao Jinliang
Student ID: 2022141450043

C++ Programming
Assignment 2

Section 1 The usage of CMake

Part1 The Complementary of the original code.

main.cpp

```cpp
#include <iostream>
#include <cstring>
#include "stuinfo.h"
using namespace std;



int main(){
    int n;
    stuinfo stu[10];
    cout << "Please make sure the amount of the students is: ";
    cin >> n;
    inputstu(stu, n);
    cout << "The information of the students is as below: " << endl;
    showstu(stu, n);
    sortstu(stu, n);
    cout << "The information of the students after sorting is as below: " << endl;
    showstu(stu, n);
    char ch[20];
    cout << "Please input the name of the student you want to find: ";
    cin >> ch;
    if(findstu(stu, n, ch)) {
        cout << "The student is in the list." << endl;
    } else {
        cout << "The student is not in the list." << endl;
    }
    return 0;
}
```

stuinfo.h

```cpp
//
```

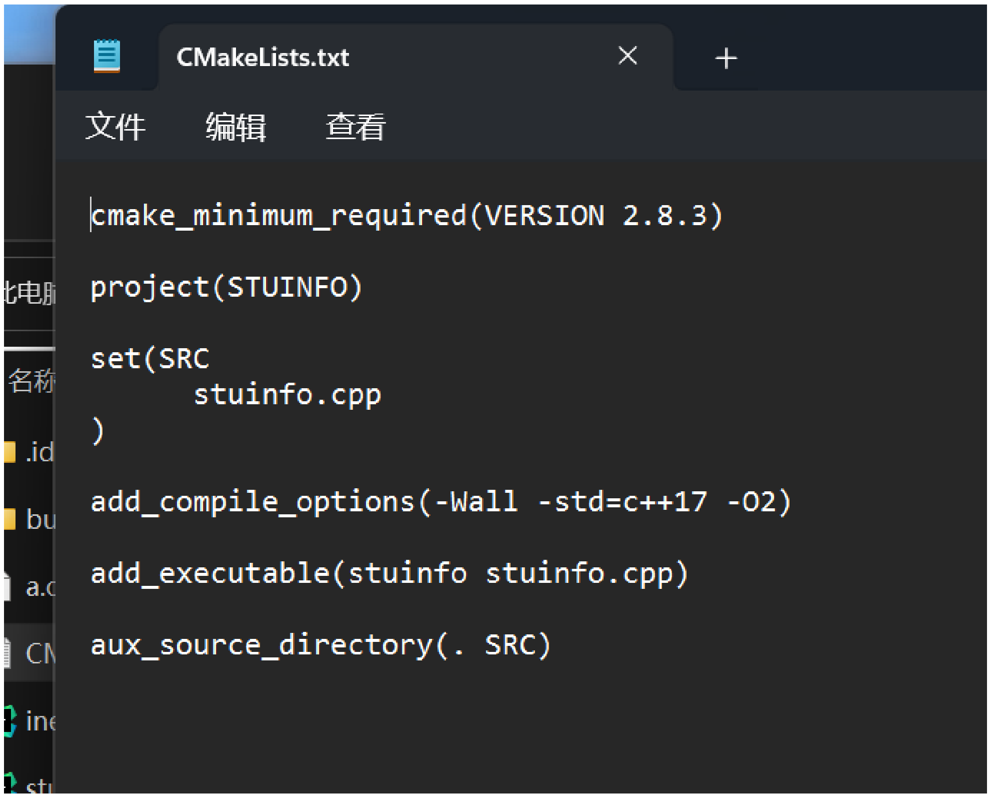
```cpp
// Created by GAO on 2023/3/10.
//

#ifndef STUINFO_STUS_Y_H
#define STUINFO_STUS_Y_H

#include <iostream>
#include <cstring>

using namespace std;

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

void inputstu(stuinfo stu[] , int n){
    for(int i = 0; i < n; i++) {
        cout << "Please input the name of student" << i+1 << ": ";
        cin >> stu[i].name;
        for(int j = 0; j < 3; j++) {
            cout << "The score" << j+1 << " for " << stu[i].name << ": ";
            cin >> stu[i].score[j];
        }
        stu[i].ave = (stu[i].score[0] + stu[i].score[1] + stu[i].score[2]) / 3.0;
    }
}
// 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){
    for(int i = 0; i < n; i++) {
        cout << "The name of student" << i+1 << " is: " << stu[i].name << endl;
        for(int j = 0; j < 3; j++) {
            cout << "The score" << j+1 << " for " << stu[i].name << "is : " << stu[i].score[j] << endl;
        }
        cout << "The average score for " << stu[i].name << "is : " << stu[i].ave << endl;
    }
} //displays the contents of the structure, one student one line.
void sortstu(stuinfo stu[] , int n){
    stuinfo temp;
    for(int i = 0; i < n; i++) {
        for(int j = i+1; j < n; j++) {
            if(stu[i].ave < stu[j].ave) {
                temp = stu[i];
                stu[i] = stu[j];
                stu[j] = temp;
            }
        }
    }
}//sorts in descending order of average of three scores.
bool findstu(stuinfo stu[] , int n, char ch[]){
    for(int i = 0; i < n; i++) {
        if(strcmp(stu[i].name, ch) == 0) {
            return true;
        }
    }
    return false;
} //finds if given characters is the student’ s name.



#endif //STUINFO_STUS_Y_H
```

```cpp
#include <iostream>
#include <cstring>
#include "stuinfo.h"
using namespace std;



int main(){
    int n;
    stuinfo stu[10];
    cout << "Please make sure the amount of the students is: ";
    cin >> n;
    inputstu(stu, n);
    cout << "The information of the students is as below: " << endl;
    showstu(stu, n);
    sortstu(stu, n);
    cout << "The information of the students after sorting is as below: " << endl;
    showstu(stu, n);
    char ch[20];
    cout << "Please input the name of the student you want to find: ";
    cin >> ch;
    if(findstu(stu, n, ch)) {
        cout << "The student is in the list." << endl;
    } else {
        cout << "The student is not in the list." << endl;
    }
    return 0;
}
```

stuinfo.h

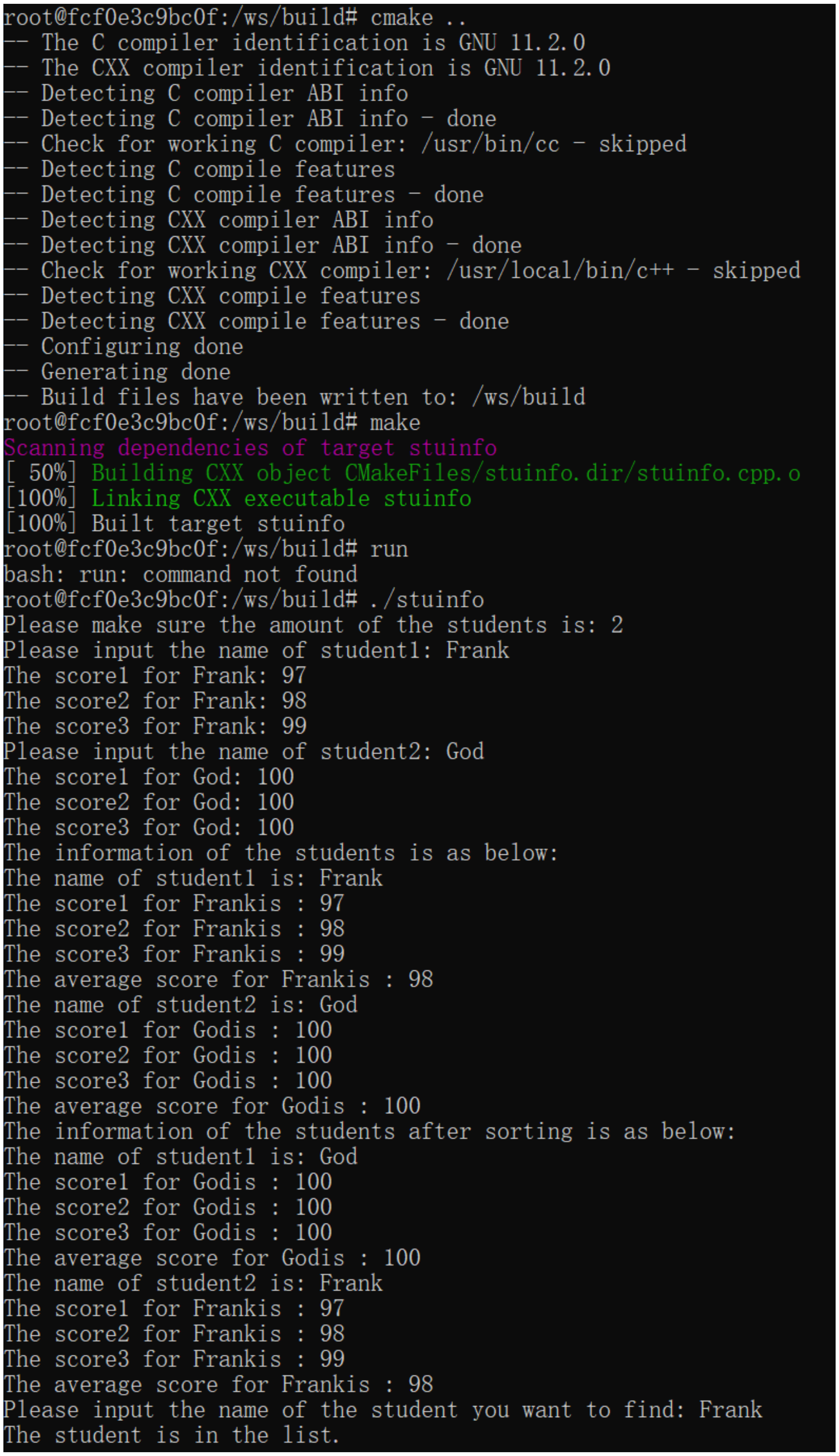
```cpp
//
// Created by GAO on 2023/3/10.
//

#ifndef STUINFO_STUS_Y_H
#define STUINFO_STUS_Y_H

#include <iostream>
#include <cstring>

using namespace std;

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

void inputstu(stuinfo stu[] , int n){
    for(int i = 0; i < n; i++) {
        cout << "Please input the name of student" << i+1 << ": ";
        cin >> stu[i].name;
        for(int j = 0; j < 3; j++) {
            cout << "The score" << j+1 << " for " << stu[i].name << ": ";
            cin >> stu[i].score[j];
        }
        stu[i].ave = (stu[i].score[0] + stu[i].score[1] + stu[i].score[2]) / 3.0;
    }
}
// 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){
    for(int i = 0; i < n; i++) {
        cout << "The name of student" << i+1 << " is: " << stu[i].name << endl;
        for(int j = 0; j < 3; j++) {
            cout << "The score" << j+1 << " for " << stu[i].name << "is : " << stu[i].score[j] << endl;
        }
        cout << "The average score for " << stu[i].name << "is : " << stu[i].ave << endl;
    }
} //displays the contents of the structure, one student one line.
void sortstu(stuinfo stu[] , int n){
    stuinfo temp;
    for(int i = 0; i < n; i++) {
        for(int j = i+1; j < n; j++) {
            if(stu[i].ave < stu[j].ave) {
                temp = stu[i];
                stu[i] = stu[j];
                stu[j] = temp;
            }
        }
    }
}//sorts in descending order of average of three scores.
bool findstu(stuinfo stu[] , int n, char ch[]){
    for(int i = 0; i < n; i++) {
        if(strcmp(stu[i].name, ch) == 0) {
            return true;
        }
    }
    return false;
} //finds if given characters is the student' s name.



#endif //STUINFO_STUS_Y_H
```

Part2 The content of CMakeLists.txt.



```
cmake_minimum_required(VERSION 2.8.3)

project(STUINFO)

set(SRC
        stuinfo.cpp
)

add_compile_options(-Wall -std=c++17 -O2)

add_executable(stuinfo stuinfo.cpp)

aux_source_directory(. SRC)
```

Part3 The screenshot of the program at runtime.



```
root@fcf0e3c9bc0f:/ws/build# cmake ..
-- The C compiler identification is GNU 11.2.0
-- The CXX compiler identification is GNU 11.2.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- 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/build
root@fcf0e3c9bc0f:/ws/build# make
Scanning dependencies of target stuinfo
[ 50%] Building CXX object CMakeFiles/stuinfo.dir/stuinfo.cpp.o
[100%] Linking CXX executable stuinfo
[100%] Built target stuinfo
root@fcf0e3c9bc0f:/ws/build# run
bash: run: command not found
root@fcf0e3c9bc0f:/ws/build# ./stuinfo
Please make sure the amount of the students is: 2
Please input the name of student1: Frank
The score1 for Frank: 97
The score2 for Frank: 98
The score3 for Frank: 99
Please input the name of student2: God
The score1 for God: 100
The score2 for God: 100
The score3 for God: 100
The information of the students is as below:
The name of student1 is: Frank
The score1 for Frankis : 97
The score2 for Frankis : 98
The score3 for Frankis : 99
The average score for Frankis : 98
The name of student2 is: God
The score1 for Godis : 100
The score2 for Godis : 100
The score3 for Godis : 100
The average score for Godis : 100
The information of the students after sorting is as below:
The name of student1 is: God
The score1 for Godis : 100
The score2 for Godis : 100
The score3 for Godis : 100
The average score for Godis : 100
The name of student2 is: Frank
The score1 for Frankis : 97
The score2 for Frankis : 98
The score3 for Frankis : 99
The average score for Frankis : 98
Please input the name of the student you want to find: Frank
The student is in the list.
```

Section 2 Types

Part1 Q&A questions

Question 1 What is the usage and function of static?

Usually ”static “is one kind of symbol that modifies the variable or methods in C++.

As for static variables, we need to make sure that the declaration of it is in the form of ”static variable_type variable_name” like ”static int x” and the initialization of it is in the form of ”variable_type variable_name = value”.
And it has several functions as follows:
1.They can be used to keep track of events that occur in a function or class.
2.They can be used to store and manage data that needs to be accessed throughout the program for they are shared by all objects.
3.They can be used to manage shared resources, such as open files, database connections, etc. These resources need to be shared by multiple objects or functions and need to be released at the end of the program.

As for static methods,it is a method that belongs to a class rather than an instance of that class.It is declared with the ”static” keyword in the method signature and can access only other static members of the class.
And it also has some functions,one of them as follows:
It can be used to implement public methods that do not require access to instance variables or instance methods of the class.

Question 2 What is implicit conversion, and how to eliminate implicit conversion?

Implicit conversion is the process of automatically converting between one data type and another without the need for an explicit type conversion operation. This conversion is usually done automatically by the programming language, and depending on the rules of the different programming languages, it is possible to implicitly convert some data types to some other data types.

In order to eliminate implicit conversions we can use these methods below:
1.If possible, we can use explicit conversions to convert one data type to another.
2.We can use type-checking code to ensure that unreasonable data type conversions do not occur.
3.We can develop a good programming habit, specifying the range and value of each data type in the program.

Part2 Program explanation questions

File1

```cpp
#include <iostream>

using namespace std;
int main() {

    int num1 = 1234567890;
    int num2 = 1234567890;
    int sum = num1 + num2;
    cout << "sum = " << sum << endl;
    //int型变量的范围是-2147483648~2147483647, 而num1和num2的和超出了这个范围, 溢出了

    float f1 = 1234567890.0f;
    float f2 = 1.0f;
    float fsum = f1 + f2;
    cout << "fsum = " << fsum << endl;
    cout << "(fsum == f1) is " << (fsum == f1) << endl;
    /*float型数据是有有效位数的, 所以f1+f2的结果并不是f1的值, 而是f1的近似值, 在保留6位有效数字时,
    1.0无法对f1的6位有效数字做出影响, 所以f1和fsum保留的是一个值, 才会出现二者相等的情况*/

    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;
    //这里体现出了浮点数的精度损失
    //10个0.1f相加时, 由于每个浮点数的表示存在舍入误差, 这些误差会在运算过程中积累, 导致最终结果与1略有
        偏差
    //0.1f*10时, 仅涉及一次乘法运算, 精度损失较小, 结果可能更接近于1

    return 0;
}
```

File2

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

int main()
{
    cout << fixed;
    float f1 = 1.0f;
    cout<<"f1 = "<<f1<<endl;
    //float的精度损失在此不会影响f1保留六位有效数的值

    float a = 0.1f;
    float f2 = a+a+a+a+a+a+a+a+a+a;
    cout<<"f2 = "<<f2<<endl;
    //float的精度损失在此不会影响f2保留六位有效数的值

    if(f1 == f2)
    cout << "f1 = f2" << endl;
    else
    cout << "f1 != f2" << endl;
    //float在比较时，是按照原值，不会保留六位有效数字运算，并且f1和f2的精度损失不一样，所以二者不相等
    return 0;
}
```

File3

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

int main()
{
    int a, b;
    double c, d;

    a = 19.99 + 21.99;
    //因为a是int型，19.99+21.99优先运算成41.98再隐式转换成int型，取整成41，赋给a
    b = (int)19.99 + (int)21.99;
    //因为19.99和21.99都被优先强制转换成int型的19和21，再相加，结果为40，赋给b
    c = 23 / 8;
    //23/8优先运算成2再赋值给c，于是才会如此输出
    d = 23 / 8.0;
    //23/8.0优先运算成2.875（在该除法中，由于分母是double型，分子被隐式转换为double型，所以运算可以保留
        小数）再赋值给d

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

Section 3 Structs

Part1 Please try to give an explanation of the situations in which alignas take effect and fail based on the above program fragment.

```
//使用一串数字表示存储空间，每个数字表示一个字节，1表示占用，0表示空闲
// alignas 生效的情况
struct Info {
    uint8_t a;//一个字节
    uint16_t b;//两个字节
    uint8_t c;//一个字节
};
std::cout << sizeof(Info) << std::endl;// 6 2 + 2 + 2
std::cout << alignof(Info) << std::endl;// 2
//存储空间表示为（10 11 10）1（一个字节存放a）0（由于b得按2的倍数个字节读取，所以此处空闲）11（两个
    字节存放b）1（存放c）0
//需要注意的是，alignas只能保证存储空间的对齐，并且只能对齐到max(sizeof(a),sizeof(b), sizeof(c))的正
    整数倍数，因此下面一例有效

struct alignas(4) Info2 {
    uint8_t a;//一个字节
    uint16_t b;//两个字节
    uint8_t c;//一个字节
};
std::cout << sizeof(Info2) << std::endl;// 8 4 + 4
std::cout << alignof(Info2) << std::endl;// 4
//存储空间表示为（1011 1000）1（一个字节存放a）0（由于b得按2的倍数个字节读取，所以此处空闲）11（两个
    字节存放b）1（存放c）000（由于Info2的对齐为4，所以此处空闲）

// alignas 失效的情况
struct Info {
    uint8_t a;//一个字节
    uint32_t b;//四个字节
    uint8_t c;//一个字节
};
std::cout << sizeof(Info) << std::endl;// 12 4 + 4 + 4
std::cout << alignof(Info) << std::endl; // 4
//存储空间表示为（1000 1111 1000）1（一个字节存放a）000（由于b得按4的倍数个字节读取，所以此处3个空
    闲）1111（四个字节存放b）1（存放c）000（由于Info的对齐为4，所以此处空闲）
struct alignas(2) Info2 {
    uint8_t a;//一个字节
    uint32_t b;//四个字节
    uint8_t c;//一个字节
};
std::cout << sizeof(Info2) << std::endl;// 12 4 + 4 + 4
std::cout << alignof(Info2) << std::endl;// 4
```

```
    //存储空间表示为 (1000 1111 1000) 1 (一个字节存放a) 000 (由于b得按4的倍数个字节读取，所以此处3个空
        闲) 1111 (四个字节存放b) 1 (存放c) 000 (由于Info2的对齐为4，所以此处空闲)
    //上文说过alignas只能对齐到max(sizeof(a),sizeof(b), sizeof(c))的正整数倍数，因此Info2的对齐为4，所以
        alignas(2)失效
```

Part2 You will use any three points in the two-dimensional plane, and if these three points can form a triangle, you are asked to calculate the center of gravity, the outer center, the inner center, and the hypocenter. If the triangle cannot be formed, please give a warning. You need to compile these functions into the shared library ”lib.so”, use the shared library and write a simple program to call these four functions. You need to compile these functions into the shared library ”lib.so”, use the shared library and write a simple program to call these four functions and display the results.

1. The code is as follows.

main.cpp

```cpp
#include "Triangle1.h"
#include <iostream>
using namespace std;
int main() {
    Point A;
    Point B;
    Point C;
    cout << "Please input 3 points." << endl;
    cout << "Point A: ";
    cin >> A.x >> A.y;
    cout << "Point B: ";
    cin >> B.x >> B.y;
    cout << "Point C: ";
    cin >> C.x >> C.y;
    if(!isTriangle(A, B, C)){
        cout << "This is not a triangle." << endl;
        return 0;
    }
    cout << "The barycenter is: (" << barycenter(A, B, C).second.x << ", " << barycenter(A, B, C).second
        .y << ")" << endl;
    cout << "The circumcenter is: (" << circumcenter(A, B, C).second.x << ", " << circumcenter(A, B, C).
        second.y << ")" << endl;
    cout << "The incenter is: (" << incenter(A, B, C).second.x << ", " << incenter(A, B, C).second.y <<
        ")" << endl;
    cout << "The orthocenter is: (" << orthocenter(A, B, C).second.x << ", " << orthocenter(A, B, C).
        second.y << ")" << endl;
    return 0;
}
```

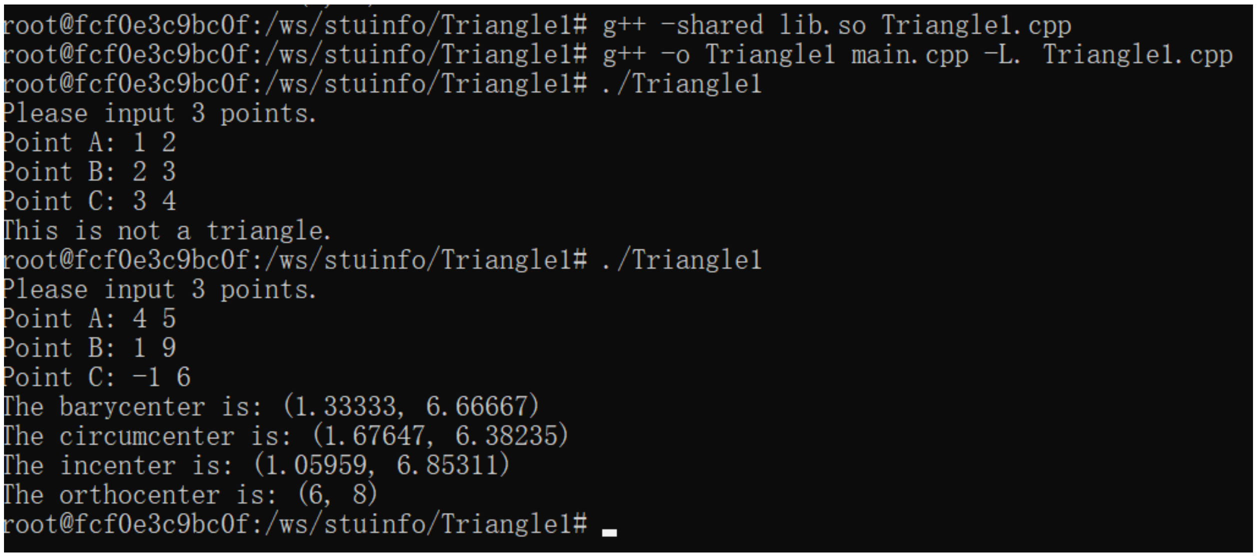
Triangle1.cpp

```
//
// Created by GAO on 2023/3/10.

#include "Triangle1.h"

bool isTriangle(const Point& A, const Point& B, const Point& C){
    if((A.x-B.x)*(B.y-C.y)==(A.y-B.y)*(B.x-C.x))return false;
    else return true;
}
// 三角形的重心
std::pair<bool, Point> barycenter(const Point& A, const Point& B, const Point& C){
    return {true, {(A.x + B.x + C.x) / 3, (A.y + B.y + C.y) / 3}};
}
// 三角形的外心
std::pair<bool, Point> circumcenter(const Point& A, const Point& B, const Point& C){
    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;
    double x=((c1*b2)-(c2*b1))/((a1*b2)-(a2*b1));
    double y=((a1*c2)-(a2*c1))/((a1*b2)-(a2*b1));
    return {true, {x, y}};
}
// 三角形的内心
std::pair<bool, Point> incenter(const Point& A, const Point& B, const Point& C){
    double a = sqrt((B.x-C.x)*(B.x-C.x)+(B.y-C.y)*(B.y-C.y));
    double b = sqrt((A.x-C.x)*(A.x-C.x)+(A.y-C.y)*(A.y-C.y));
    double c = sqrt((B.x-A.x)*(B.x-A.x)+(B.y-A.y)*(B.y-A.y));
    return {true, {(a*A.x+b*B.x+c*C.x)/(a+b+c),(a*A.y+b*B.y+c*C.y)/(a+b+c)}};
}
// 三角形的垂心
std::pair<bool, Point> orthocenter(const Point& A, const Point& B, const Point& C){
    double a1=B.x-C.x;
    double b1=B.y-C.y;
    double c1=a1*A.y-b1*A.x;

    double a2=A.x-C.x;
    double b2=A.y-C.y;
    double c2=a2*B.y-b2*B.x;

    double x=(a1*c2-a2*c1)/(a2*b1-a1*b2);
    double y=(b1*c2-b2*c1)/(a2*b1-a1*b2);

    return {true, {x, y}};
}
```

Triangle1.h

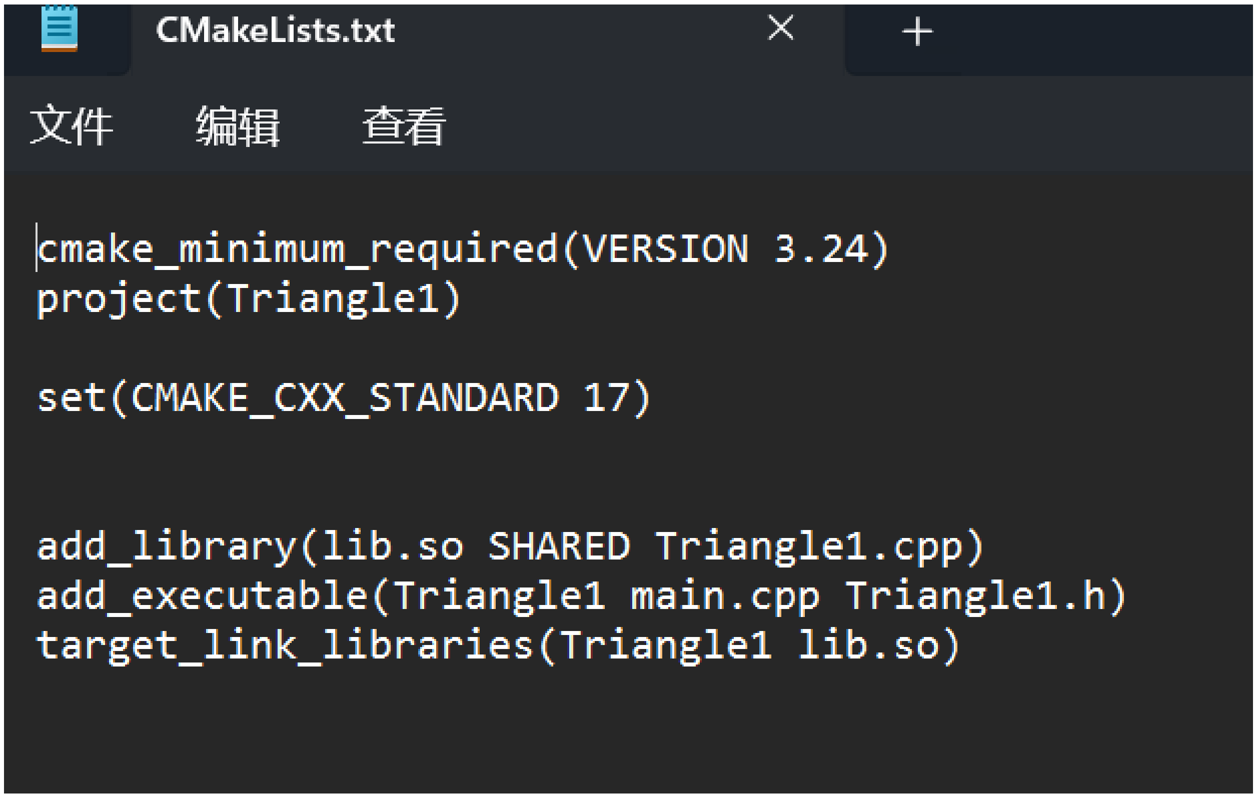
```
//
// Created by GAO on 2023/3/10.
//

#ifndef TRIANGLE1_H
#define TRIANGLE1_H
#include <cmath>

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 = {O, {1, 0}}, Oy = {O, {0, 1}}; // 坐标轴
const double PI = std::acos(-1), EPS = 1e-9; //PI 与偏差值

bool isTriangle(const Point& A, const Point& B, const Point& C);
// 三角形的重心
std::pair<bool, Point> barycenter(const Point& A, const Point& B, const Point& C);
// 三角形的外心
std::pair<bool, Point> circumcenter(const Point& A, const Point& B, const Point& C);
// 三角形的内心
std::pair<bool, Point> incenter(const Point& A, const Point& B, const Point& C);
// 三角形的垂心
std::pair<bool, Point> orthocenter(const Point& A, const Point& B, const Point& C);

#endif //TRIANGLE1_TRIANGLE1_H
```

2. The function is called as follows.



```
root@fcf0e3c9bc0f:/ws/stuinfo/Triangle1# g++ -shared lib.so Triangle1.cpp
root@fcf0e3c9bc0f:/ws/stuinfo/Triangle1# g++ -o Triangle1 main.cpp -L. Triangle1.cpp
root@fcf0e3c9bc0f:/ws/stuinfo/Triangle1# ./Triangle1
Please input 3 points.
Point A: 1 2
Point B: 2 3
Point C: 3 4
This is not a triangle.
root@fcf0e3c9bc0f:/ws/stuinfo/Triangle1# ./Triangle1
Please input 3 points.
Point A: 4 5
Point B: 1 9
Point C: -1 6
The barycenter is: (1.33333, 6.66667)
The circumcenter is: (1.67647, 6.38235)
The incenter is: (1.05959, 6.85311)
The orthocenter is: (6, 8)
root@fcf0e3c9bc0f:/ws/stuinfo/Triangle1# _
```

3. The CMakeLists file is as follows.



```
cmake_minimum_required(VERSION 3.24)
project(Triangle1)

set(CMAKE_CXX_STANDARD 17)


add_library(lib.so SHARED Triangle1.cpp)
add_executable(Triangle1 main.cpp Triangle1.h)
target_link_libraries(Triangle1 lib.so)
```

Section 4 C++ Dynamic Memory Requests

Part1 Please indicate where the data in (1)(2)(3)(4)(5)(6) are located in memory.

```
#include <iostream>

class A {
    int num;
};

static int a; //(1)
//静态变量a存放在内存的全局/静态存储区，程序结束时由系统释放。
auto b=0; //(2)
//自动变量b被存储在栈区域。
int main()
{
char s[]="abc"; //(3)
//字符数组s被存储在栈区域。
char *p="123456"; //(4)。
//字符指针p被存储在栈区域，指向的字符串常量"123456"被存储在常量存储区。
p1= (char *)malloc(10); // (5)
//动态分配的字符指针p1被存储在栈区域，指向的10个字节的内存块被分配在堆区域。
A *a = new A(); // (6)
//通过new动态分配的类A对象指针a被存储在栈区域，指向的A对象被存储在堆区域。
}
```

Part2 Q&A questions

1. The difference between new and malloc.

(1) 申请内存的方式不同：new 是一个运算符，malloc 是一个函数。
(2) 分配的内存大小计算方式不同：new 的操作数是类型，申请内存时会自动计算类型大小；malloc 的操作数是要分配的字节数。
(3) 返回值类型不同：new 返回的是分配的对象的指针，可以直接用于调用该对象的成员函数；malloc 返回的是 void 类型指针，需要进行类型转换后才能使用。
(4) 能否执行对象构造函数：new 在分配内存后会调用对象的构造函数进行初始化，而 malloc 只分配内存，不执行构造函数。
(5) 能否自动释放分配的内存：new 分配的内存会在对象被删除时自动释放；而 malloc 分配的内存需要手动使用 free() 函数进行释放。
(6)new 更加面向对象，适合用于申请单个对象的内存空间，而 malloc 更加通用，适合用于申请大块的内存空间。

2. What kind of functions do delete p,delete[ ] p and allocator have?

(1)delete p: 用于释放用 new 分配的单个对象的内存，其中 p 是指向要释放的对象的指针。使用 delete 会自动调用该对象的析构函数，然后释放分配给该对象的内存空间。

(2)delete[] p: 用于释放用 new[] 分配的数组对象的内存，其中 p 是指向要释放的数组对象的指针。使用 delete[] 会自动调用每个元素的析构函数，然后释放分配给该数组的内存空间。
(3)allocator：是一个标准库类模板，用于在动态内存中分配对象。它可以根据不同的内存池实现，提供了与 new、delete 操作类似的内存分配方式。与 new 相比，allocator 的优势在于更加灵活，可以通过更改内存池来优化内存分配的效率。与 malloc 相比，allocator 具有更好的类型安全性和内存管理能力。

3. Can the storage space requested by malloc be freed by delete?

不能，因为 malloc 和 new 分别使用了不同的内存管理机制，它们的内部实现方式也不同。使用 new 申请的内存空间必须使用 delete 进行释放，因为 new 和 delete 都是 C++ 标准库提供的内存管理工具，它们遵循相同的内存管理机制。而 malloc 和 free 则是 C 标准库提供的内存管理工具，它们与 new/delete 的内存管理机制不同。

4. How do malloc and free work?

(1)malloc 函数首先会检查当前内存池中是否有足够的空闲内存可用于分配所需大小的内存块，如果有，则直接从内存池中分配出一块内存空间，并将这块内存空间的起始地址返回给调用者。
(2) 如果内存池中没有足够的空闲内存可用，则 malloc 函数会调用系统调用（例如 sbrk 或 mmap）向操作系统请求更多的内存空间。操作系统将分配一块物理内存，并将其映射到调用进程的虚拟地址空间中，然后将这块内存空间的起始地址返回给 malloc 函数。
(3)malloc 函数将这块内存空间分割成大小合适的内存块，将其中一块分配给调用者，并将剩余的内存块加入到内存池中，以备后续使用。
(4)free 函数将调用者传入的内存块地址返回给内存池，并将这个内存块标记为可用状态，以便后续的 malloc 函数可以重复利用这个内存块。

Part3 Exercise

1. The program code is as follows.

main.cpp

```cpp
#include "RandomGenerator.h"

using namespace std;

int main() {
    int n;
    cout << "Please input the number of random numbers: ";
    cin >> n;
    int *arr = new int[n];
    default_random_engine e;
    uniform_int_distribution<int> k(0, n-1);
    for(int i = 0; i < n; i++) {
        arr[i] = k(e);
    }
    Max(arr, n);
    delete [] arr;
    return 0;
}
```

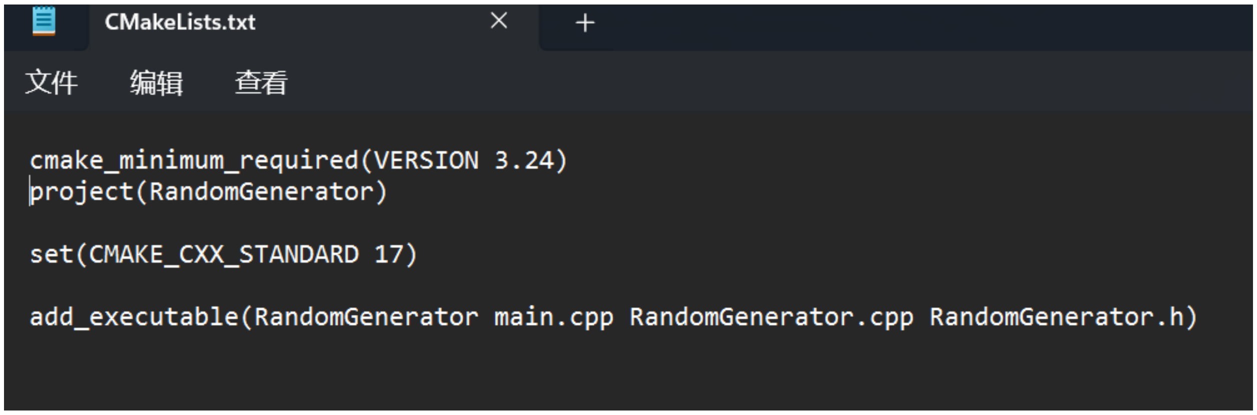
RandomGenerator.cpp

```
//
// Created by GAO on 2023/3/14.
//
#include "RandomGenerator.h"

void Max(int arr[], int n) {
    int max = arr[0];
    for(int i = 1; i < n; i++) {
        if(arr[i] > max) {
            max = arr[i];
        }
    }
    cout << max << endl;
}
```

RandomGenerator.h

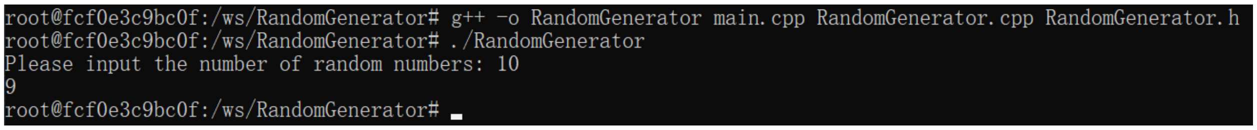
```
//
// Created by GAO on 2023/3/14.
//

#ifndef RANDOMGENERATOR_RANDOMGENERATOR_H
#define RANDOMGENERATOR_RANDOMGENERATOR_H

#include <iostream>
#include <random>

using namespace std;

void Max(int arr[], int n);

#endif //RANDOMGENERATOR_RANDOMGENERATOR_H
```

2. The CMakeList file is as follows.



```
cmake_minimum_required(VERSION 3.24)
project(RandomGenerator)

set(CMAKE_CXX_STANDARD 17)

add_executable(RandomGenerator main.cpp RandomGenerator.cpp RandomGenerator.h)
```

3. The running result is as follows.



```
root@fcf0e3c9bc0f:/ws/RandomGenerator# g++ -o RandomGenerator main.cpp RandomGenerator.cpp RandomGenerator.h
root@fcf0e3c9bc0f:/ws/RandomGenerator# ./RandomGenerator
Please input the number of random numbers: 10
9
root@fcf0e3c9bc0f:/ws/RandomGenerator# _
```

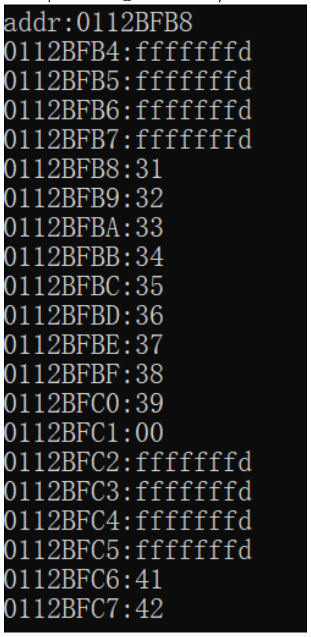
Section 5 Debug & Release

Part1 In VS2022 in x86/Debug, Release mode with Linux running under Debug with GDB.

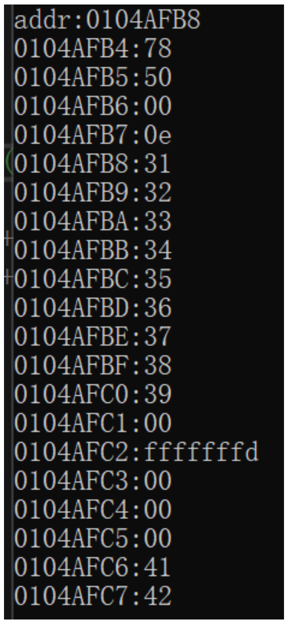
Results

I.(1)(2)(3) All comments, observe the running results

x86/debug & x86/release

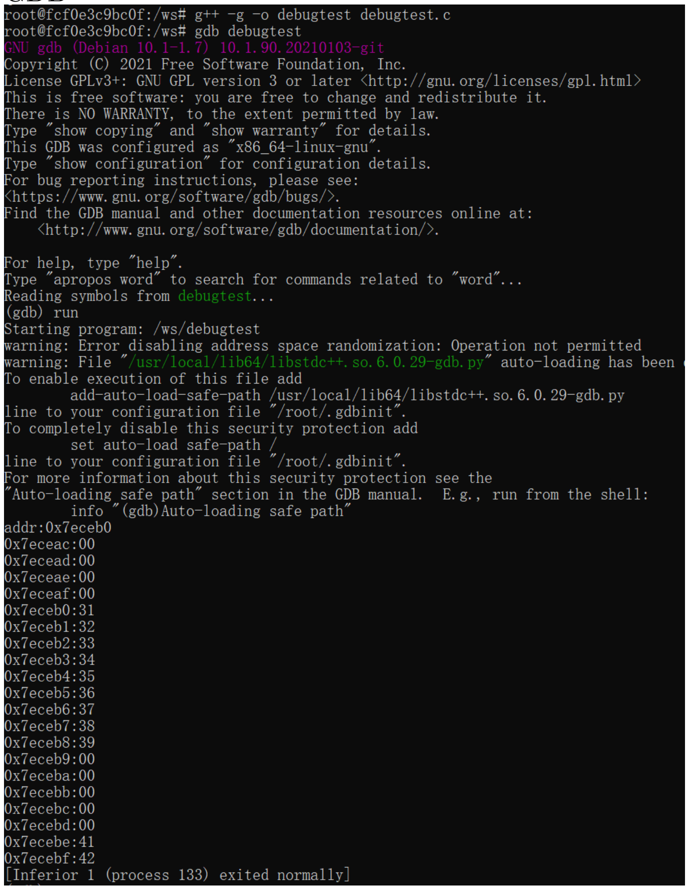


```
addr:0112BFB8
0112BFB4:fffffffd
0112BFB5:fffffffd
0112BFB6:fffffffd
0112BFB7:fffffffd
0112BFB8:31
0112BFB9:32
0112BFBA:33
0112BFBB:34
0112BFBC:35
0112BFBD:36
0112BFBE:37
0112BFBF:38
0112BFC0:39
0112BFC1:00
0112BFC2:fffffffd
0112BFC3:fffffffd
0112BFC4:fffffffd
0112BFC5:fffffffd
0112BFC6:41
0112BFC7:42
```



```
addr:0104AFB8
0104AFB4:78
0104AFB5:50
0104AFB6:00
0104AFB7:0e
0104AFB8:31
0104AFB9:32
0104AFBA:33
0104AFBB:34
0104AFBC:35
0104AFBD:36
0104AFBE:37
0104AFBF:38
0104AFC0:39
0104AFC1:00
0104AFC2:fffffffd
0104AFC3:00
0104AFC4:00
0104AFC5:00
0104AFC6:41
0104AFC7:42
```

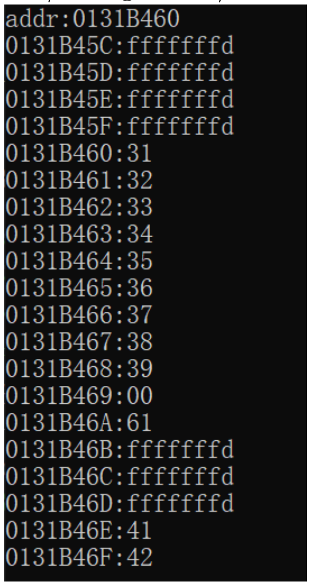
GDB



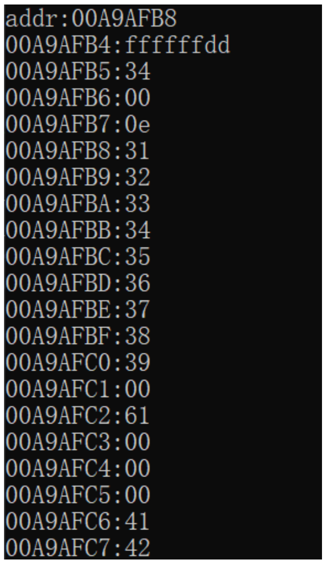
```
root@fcf0e3c9bc0f:/ws# g++ -g -o debugtest debugtest.c
root@fcf0e3c9bc0f:/ws# gdb debugtest
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from debugtest...
(gdb) run
Starting program: /ws/debugtest
warning: Error disabling address space randomization: Operation not permitted
warning: File "/usr/local/lib64/libstdc++.so.6.0.29-gdb.py" auto-loading has been
To enable execution of this file add
        add-auto-load-safe-path /usr/local/lib64/libstdc++.so.6.0.29-gdb.py
line to your configuration file "/root/.gdbinit".
To completely disable this security protection add
        set auto-load safe-path /
line to your configuration file "/root/.gdbinit".
For more information about this security protection see the
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
        info "(gdb)Auto-loading safe path"
addr:0x7eceb0
0x7eceac:00
0x7ecead:00
0x7eceae:00
0x7eceaf:00
0x7eceb0:31
0x7eceb1:32
0x7eceb2:33
0x7eceb3:34
0x7eceb4:35
0x7eceb5:36
0x7eceb6:37
0x7eceb7:38
0x7eceb8:39
0x7eceb9:00
0x7eceba:00
0x7ecebb:00
0x7ecebc:00
0x7ecebd:00
0x7ecebe:41
0x7ecebf:42
[Inferior 1 (process 133) exited normally]
```

II.(1) release, (2)(3) comment, observe the operation results

x86/debug & x86/release

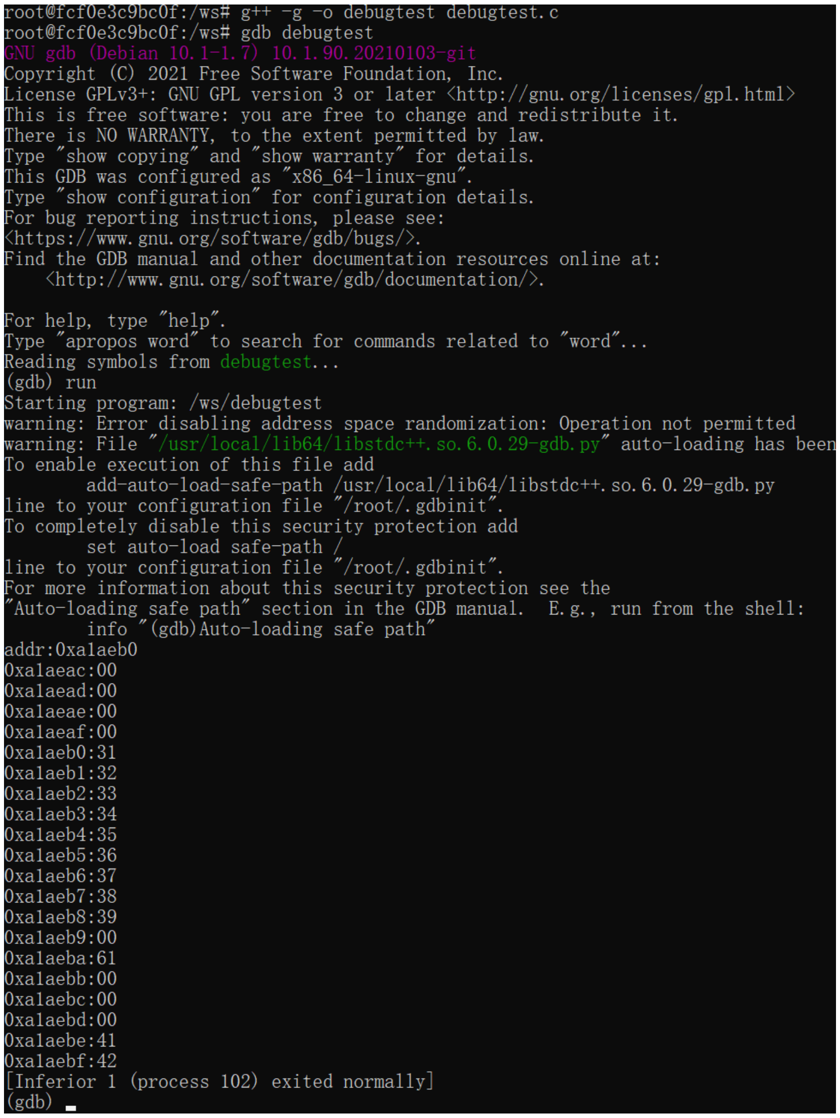


```
addr:0131B460
0131B45C:fffffffd
0131B45D:fffffffd
0131B45E:fffffffd
0131B45F:fffffffd
0131B460:31
0131B461:32
0131B462:33
0131B463:34
0131B464:35
0131B465:36
0131B466:37
0131B467:38
0131B468:39
0131B469:00
0131B46A:61
0131B46B:fffffffd
0131B46C:fffffffd
0131B46D:fffffffd
0131B46E:41
0131B46F:42
```



```
addr:00A9AFB8
00A9AFB4:ffffffdd
00A9AFB5:34
00A9AFB6:00
00A9AFB7:0e
00A9AFB8:31
00A9AFB9:32
00A9AFBA:33
00A9AFBB:34
00A9AFBC:35
00A9AFBD:36
00A9AFBE:37
00A9AFBF:38
00A9AFC0:39
00A9AFC1:00
00A9AFC2:61
00A9AFC3:00
00A9AFC4:00
00A9AFC5:00
00A9AFC6:41
00A9AFC7:42
```

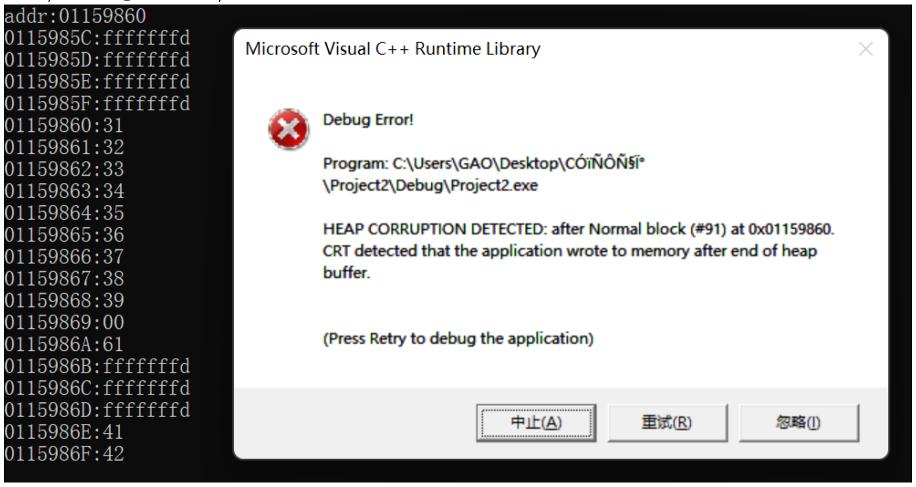
GDB



```
root@fcf0e3c9bc0f:/ws# g++ -g -o debugtest debugtest.c
root@fcf0e3c9bc0f:/ws# gdb debugtest
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from debugtest...
(gdb) run
Starting program: /ws/debugtest
warning: Error disabling address space randomization: Operation not permitted
warning: File "/usr/local/lib64/libstdc++.so.6.0.29-gdb.py" auto-loading has been
To enable execution of this file add
        add-auto-load-safe-path /usr/local/lib64/libstdc++.so.6.0.29-gdb.py
line to your configuration file "/root/.gdbinit".
To completely disable this security protection add
        set auto-load safe-path /
line to your configuration file "/root/.gdbinit".
For more information about this security protection see the
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
        info "(gdb)Auto-loading safe path"
addr:0xa1aeb0
0xa1aeac:00
0xa1aead:00
0xa1aeae:00
0xa1aeaf:00
0xa1aeb0:31
0xa1aeb1:32
0xa1aeb2:33
0xa1aeb3:34
0xa1aeb4:35
0xa1aeb5:36
0xa1aeb6:37
0xa1aeb7:38
0xa1aeb8:39
0xa1aeb9:00
0xa1aeba:61
0xa1aebb:00
0xa1aebc:00
0xa1aebd:00
0xa1aebe:41
0xa1aebf:42
[Inferior 1 (process 102) exited normally]
(gdb)
```

III.(1)(3) Release, (2) Comment, Observe run results

x86/debug & x86/release



```
addr:01159860
0115985C:fffffffd
0115985D:fffffffd
0115985E:fffffffd
0115985F:fffffffd
01159860:31
01159861:32
01159862:33
01159863:34
01159864:35
01159865:36
01159866:37
01159867:38
01159868:39
01159869:00
0115986A:61
0115986B:fffffffd
0115986C:fffffffd
0115986D:fffffffd
0115986E:41
0115986F:42
```

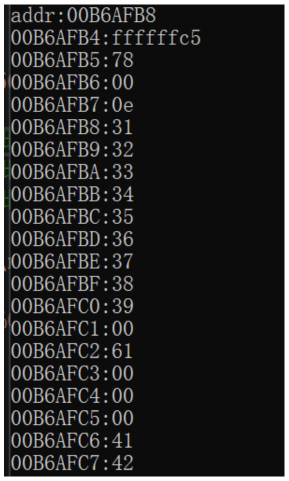
Microsoft Visual C++ Runtime Library

Debug Error!

Program: C:\Users\GAO\Desktop\CÓïÑÔÑšî°
\Project2\Debug\Project2.exe

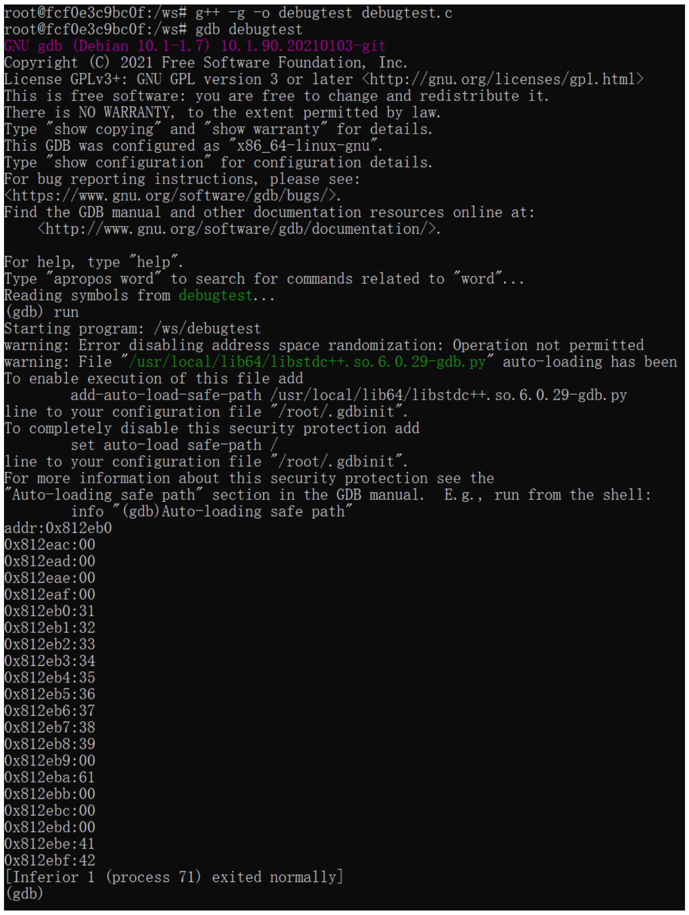
HEAP CORRUPTION DETECTED: after Normal block (#91) at 0x01159860.
CRT detected that the application wrote to memory after end of heap buffer.

(Press Retry to debug the application)



```
addr:00B6AFB8
00B6AFB4:ffffffc5
00B6AFB5:78
00B6AFB6:00
00B6AFB7:0e
00B6AFB8:31
00B6AFB9:32
00B6AFBA:33
00B6AFBB:34
00B6AFBC:35
00B6AFBD:36
00B6AFBE:37
00B6AFBF:38
00B6AFC0:39
00B6AFC1:00
00B6AFC2:61
00B6AFC3:00
00B6AFC4:00
00B6AFC5:00
00B6AFC6:41
00B6AFC7:42
```

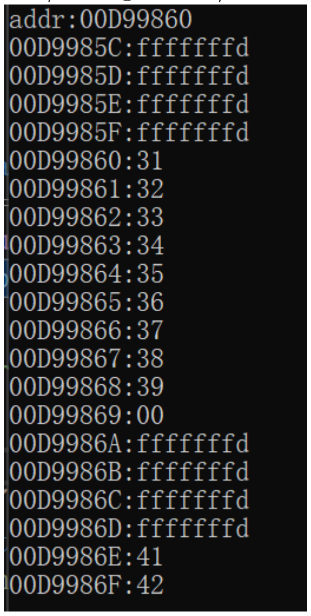
GDB



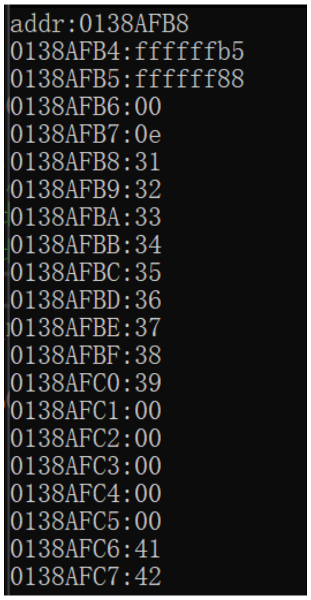
```
root@fcf0e3c9bc0f:/ws# g++ -g -o debugtest debugtest.c
root@fcf0e3c9bc0f:/ws# gdb debugtest
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from debugtest...
(gdb) run
Starting program: /ws/debugtest
warning: Error disabling address space randomization: Operation not permitted
warning: File "/usr/local/lib64/libstdc++.so.6.0.29-gdb.py" auto-loading has been
To enable execution of this file add
        add-auto-load-safe-path /usr/local/lib64/libstdc++.so.6.0.29-gdb.py
line to your configuration file "/root/.gdbinit".
To completely disable this security protection add
        set auto-load safe-path /
line to your configuration file "/root/.gdbinit".
For more information about this security protection see the
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
        info "(gdb)Auto-loading safe path"
addr:0x812eb0
0x812eac:00
0x812ead:00
0x812eae:00
0x812eaf:00
0x812eb0:31
0x812eb1:32
0x812eb2:33
0x812eb3:34
0x812eb4:35
0x812eb5:36
0x812eb6:37
0x812eb7:38
0x812eb8:39
0x812eb9:00
0x812eba:61
0x812ebb:00
0x812ebc:00
0x812ebd:00
0x812ebe:41
0x812ebf:42
[Inferior 1 (process 71) exited normally]
(gdb)
```

IV.(1)(2)(3) Release all and observe the operation results

x86/debug & x86/release

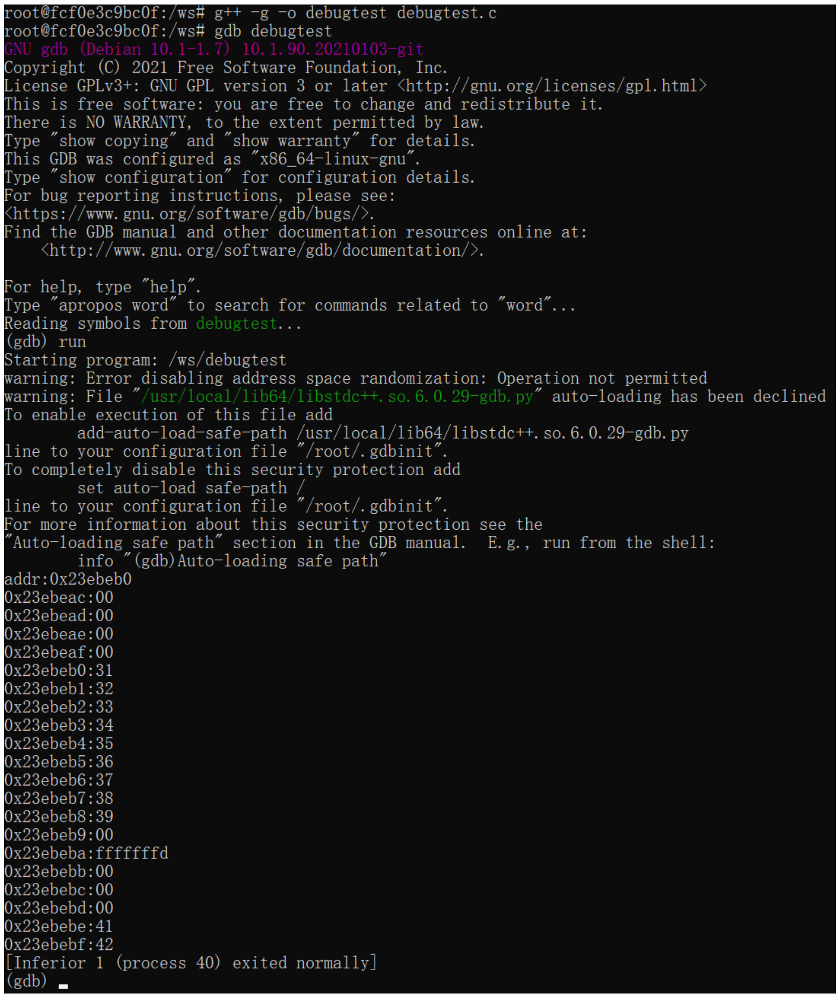


```
addr:00D99860
00D9985C:fffffffd
00D9985D:fffffffd
00D9985E:fffffffd
00D9985F:fffffffd
00D99860:31
00D99861:32
00D99862:33
00D99863:34
00D99864:35
00D99865:36
00D99866:37
00D99867:38
00D99868:39
00D99869:00
00D9986A:fffffffd
00D9986B:fffffffd
00D9986C:fffffffd
00D9986D:fffffffd
00D9986E:41
00D9986F:42
```



```
addr:0138AFB8
0138AFB4:ffffffb5
0138AFB5:ffffff88
0138AFB6:00
0138AFB7:0e
0138AFB8:31
0138AFB9:32
0138AFBA:33
0138AFBB:34
0138AFBC:35
0138AFBD:36
0138AFBE:37
0138AFBF:38
0138AFC0:39
0138AFC1:00
0138AFC2:00
0138AFC3:00
0138AFC4:00
0138AFC5:00
0138AFC6:41
0138AFC7:42
```

GDB



```
root@fcf0e3c9bc0f:/ws# g++ -g -o debugtest debugtest.c
root@fcf0e3c9bc0f:/ws# gdb debugtest
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from debugtest...
(gdb) run
Starting program: /ws/debugtest
warning: Error disabling address space randomization: Operation not permitted
warning: File "/usr/local/lib64/libstdc++.so.6.0.29-gdb.py" auto-loading has been declined
To enable execution of this file add
        add-auto-load-safe-path /usr/local/lib64/libstdc++.so.6.0.29-gdb.py
line to your configuration file "/root/.gdbinit".
To completely disable this security protection add
        set auto-load safe-path /
line to your configuration file "/root/.gdbinit".
For more information about this security protection see the
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
        info "(gdb)Auto-loading safe path"
addr:0x23ebeb0
0x23ebeac:00
0x23ebead:00
0x23ebeae:00
0x23ebeaf:00
0x23ebeb0:31
0x23ebeb1:32
0x23ebeb2:33
0x23ebeb3:34
0x23ebeb4:35
0x23ebeb5:36
0x23ebeb6:37
0x23ebeb7:38
0x23ebeb8:39
0x23ebeb9:00
0x23ebeba:fffffffd
0x23ebebb:00
0x23ebebc:00
0x23ebebd:00
0x23ebebe:41
0x23ebebf:42
[Inferior 1 (process 40) exited normally]
(gdb)
```

Analyses

VS 的 Debug 模式可以通过使用 CRT 库来判断动态申请内存访问越界 (由截图中的警告提示可以判断)。CRT 库中提供了一些宏和函数来检测内存分配和释放的情况。当发生内存越界时，VS 会提示出错的位置和原因。

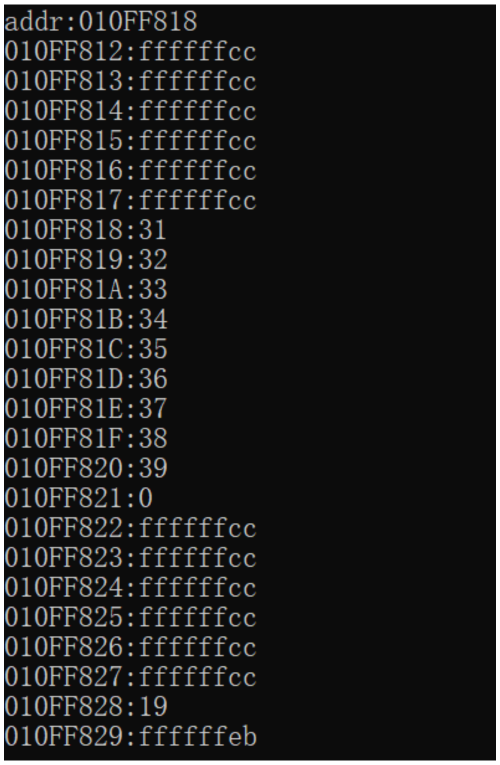
Linux 下判断动态申请内存访问越界是使用 valgrind 工具，或者使用信号处理函数来捕获 SIGSEGV 信号。

具体来说，第三次 debug 的报错以及前几次的通过显示：在 VS 中，系统会给数组的有效存储空间的前端与后端分别加上四个特殊存储单元（内容为 fffffcc)，当其被修改之后就会被判断数组越界。但是如果修改了除了这八个特殊存储单元和数组的有效存储空间的相对于数组逻辑上越界的其他存储单元的值就不会报错。特别地，如果将 fffffcc 修改为不可存储的内容，也不会报错。或许是因为，这些空间本来的意义就是不可存储吧。在 Linux 下，数组的前端与末端附近则被分别加上了四个 00 用来判断是否越界。

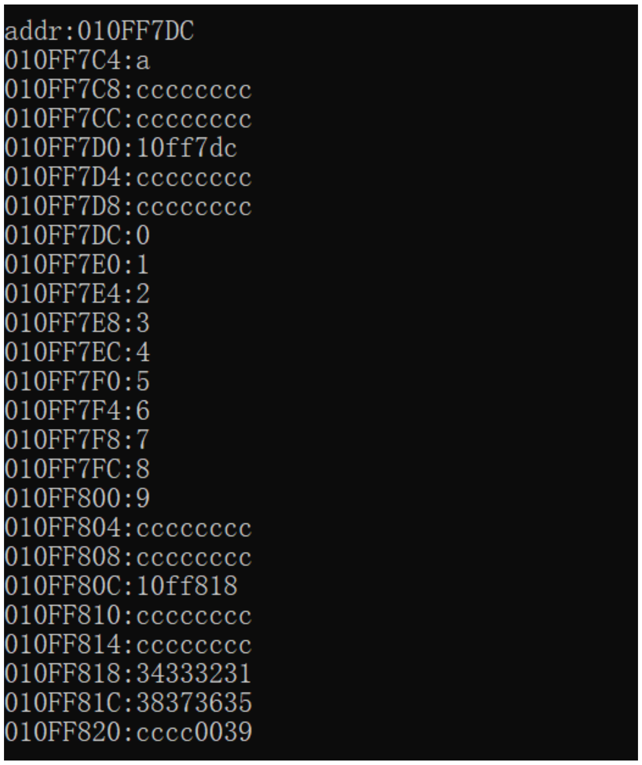
Part 2 With an understanding of the first two cases, construct your own similar program to observe the memory behavior of the array after it is out of bounds, and verify that it is similar to the dynamic application Similarity

Simply running

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cstring>

using namespace std;

int main() {
    char a[10];
    char* p = a;
    if (p == NULL) {
        return -1;
    }
    strcpy(p, "123456789");

    int b[10];
    int* q = b;
    if (q == NULL) {
        return -1;
    }
    for (int i = 0; i < 10; i++) {
        q[i] = i;
    }
    cout << "addr:" << hex << (void*)(p) << endl;
    for (int i = -6; i < 18; i++)
        cout << hex << (void*)(p + i) << ":" << int(p[i]) << endl;

    cout << endl;

    cout << "addr:" << hex << (void*)(q) << endl;
    for (int i = -6; i < 18; i++)
        cout << hex << (void*)(q + i) << ":" << int(q[i]) << endl;

    return 0;
}
```



```
addr:010FF818
010FF812:ffffffcc
010FF813:ffffffcc
010FF814:ffffffcc
010FF815:ffffffcc
010FF816:ffffffcc
010FF817:ffffffcc
010FF818:31
010FF819:32
010FF81A:33
010FF81B:34
010FF81C:35
010FF81D:36
010FF81E:37
010FF81F:38
010FF820:39
010FF821:0
010FF822:ffffffcc
010FF823:ffffffcc
010FF824:ffffffcc
010FF825:ffffffcc
010FF826:ffffffcc
010FF827:ffffffcc
010FF828:19
010FF829:ffffffeb
```

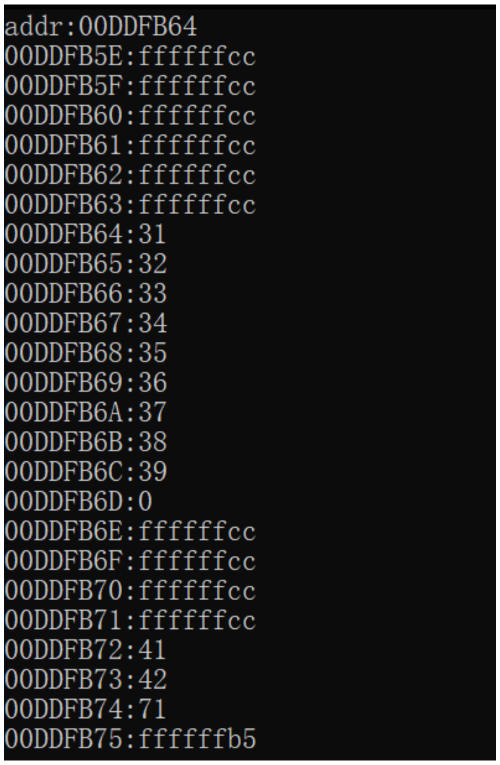


```
addr:010FF7DC
010FF7C4:a
010FF7C8:cccccccc
010FF7CC:cccccccc
010FF7D0:10ff7dc
010FF7D4:cccccccc
010FF7D8:cccccccc
010FF7DC:0
010FF7E0:1
010FF7E4:2
010FF7E8:3
010FF7EC:4
010FF7F0:5
010FF7F4:6
010FF7F8:7
010FF7FC:8
010FF800:9
010FF804:cccccccc
010FF808:cccccccc
010FF80C:10ff818
010FF810:cccccccc
010FF814:cccccccc
010FF818:34333231
010FF81C:38373635
010FF820:cccc0039
```

结果显示普通 char 型数组的前后存储了 ffffffcc 来识别是否越界，普通 int 型数组的前后存储了 cccccccc 来判断是否越界。

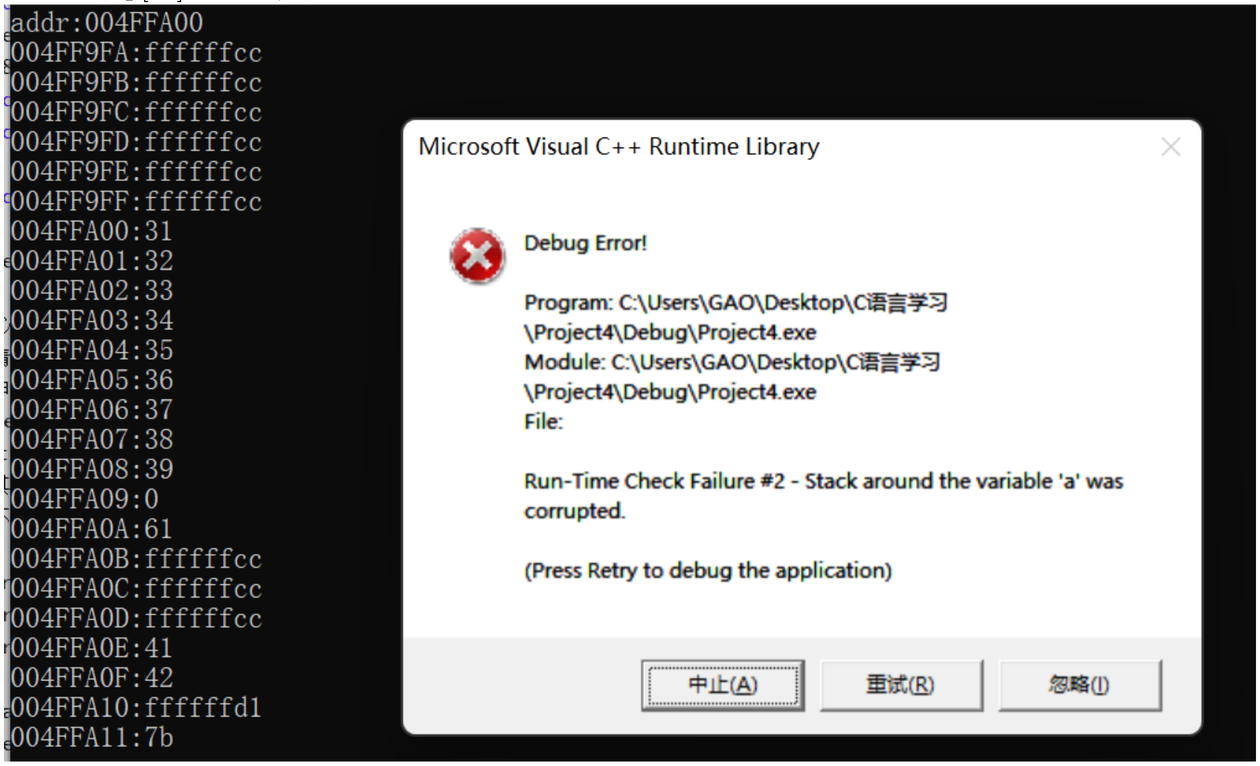
首先研究字符型数组

当我令 p[14]='A',p[15]='B' 时，运行结果如下（没有报错）



```
addr:00DDFB64
00DDFB5E:ffffffcc
00DDFB5F:ffffffcc
00DDFB60:ffffffcc
00DDFB61:ffffffcc
00DDFB62:ffffffcc
00DDFB63:ffffffcc
00DDFB64:31
00DDFB65:32
00DDFB66:33
00DDFB67:34
00DDFB68:35
00DDFB69:36
00DDFB6A:37
00DDFB6B:38
00DDFB6C:39
00DDFB6D:0
00DDFB6E:ffffffcc
00DDFB6F:ffffffcc
00DDFB70:ffffffcc
00DDFB71:ffffffcc
00DDFB72:41
00DDFB73:42
00DDFB74:71
00DDFB75:ffffffb5
```

当我再令 p[10]=’a’ 时，就报错了



```
addr:004FFA00
004FF9FA:ffffffcc
004FF9FB:ffffffcc
004FF9FC:ffffffcc
004FF9FD:ffffffcc
004FF9FE:ffffffcc
004FF9FF:ffffffcc
004FFA00:31
004FFA01:32
004FFA02:33
004FFA03:34
004FFA04:35
004FFA05:36
004FFA06:37
004FFA07:38
004FFA08:39
004FFA09:0
004FFA0A:61
004FFA0B:ffffffcc
004FFA0C:ffffffcc
004FFA0D:ffffffcc
004FFA0E:41
004FFA0F:42
004FFA10:ffffffd1
004FFA11:7b
```

Microsoft Visual C++ Runtime Library

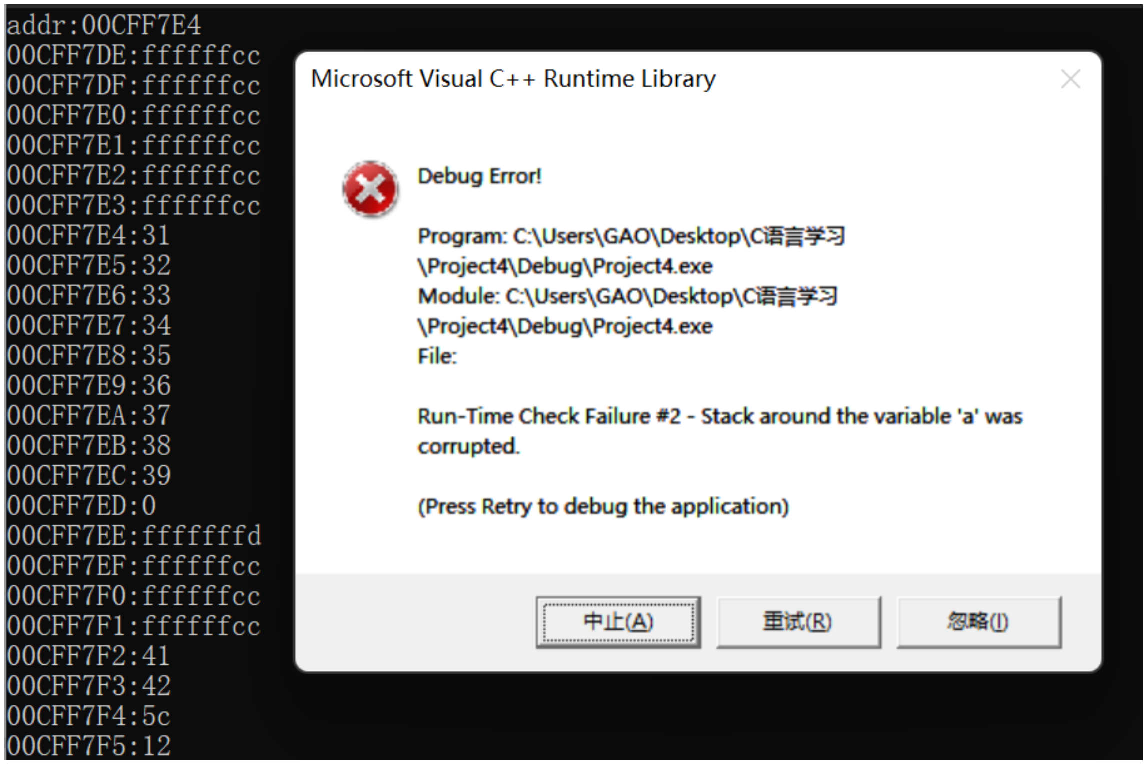
Debug Error!

Program: C:\Users\GAO\Desktop\C语言学习\Project4\Debug\Project4.exe
Module: C:\Users\GAO\Desktop\C语言学习\Project4\Debug\Project4.exe
File:

Run-Time Check Failure #2 - Stack around the variable 'a' was corrupted.

(Press Retry to debug the application)

但是当我注释掉 p[10]=’a’, 再令 p[10]=’xfd’ 时, 与动态申请不同地，报错了。所以这里规则有异

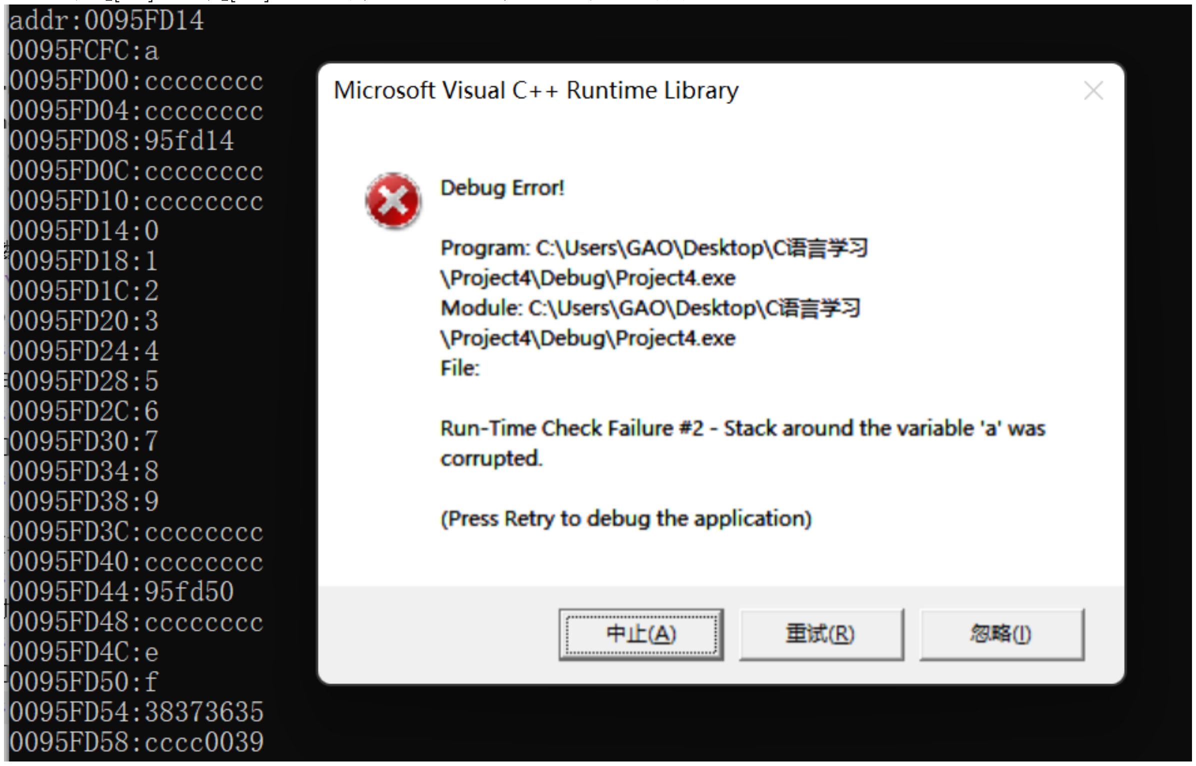


```
addr:00CFF7E4
00CFF7DE:ffffffcc
00CFF7DF:ffffffcc
00CFF7E0:ffffffcc
00CFF7E1:ffffffcc
00CFF7E2:ffffffcc
00CFF7E3:ffffffcc
00CFF7E4:31
00CFF7E5:32
00CFF7E6:33
00CFF7E7:34
00CFF7E8:35
00CFF7E9:36
00CFF7EA:37
00CFF7EB:38
00CFF7EC:39
00CFF7ED:0
00CFF7EE:fffffffd
00CFF7EF:ffffffcc
00CFF7F0:ffffffcc
00CFF7F1:ffffffcc
00CFF7F2:41
00CFF7F3:42
00CFF7F4:5c
00CFF7F5:12
```

```
Microsoft Visual C++ Runtime Library

Debug Error!

Program: C:\Users\GAO\Desktop\C语言学习
\Project4\Debug\Project4.exe
Module: C:\Users\GAO\Desktop\C语言学习
\Project4\Debug\Project4.exe
File:

Run-Time Check Failure #2 - Stack around the variable 'a' was
corrupted.

(Press Retry to debug the application)
```

其次研究整型数组

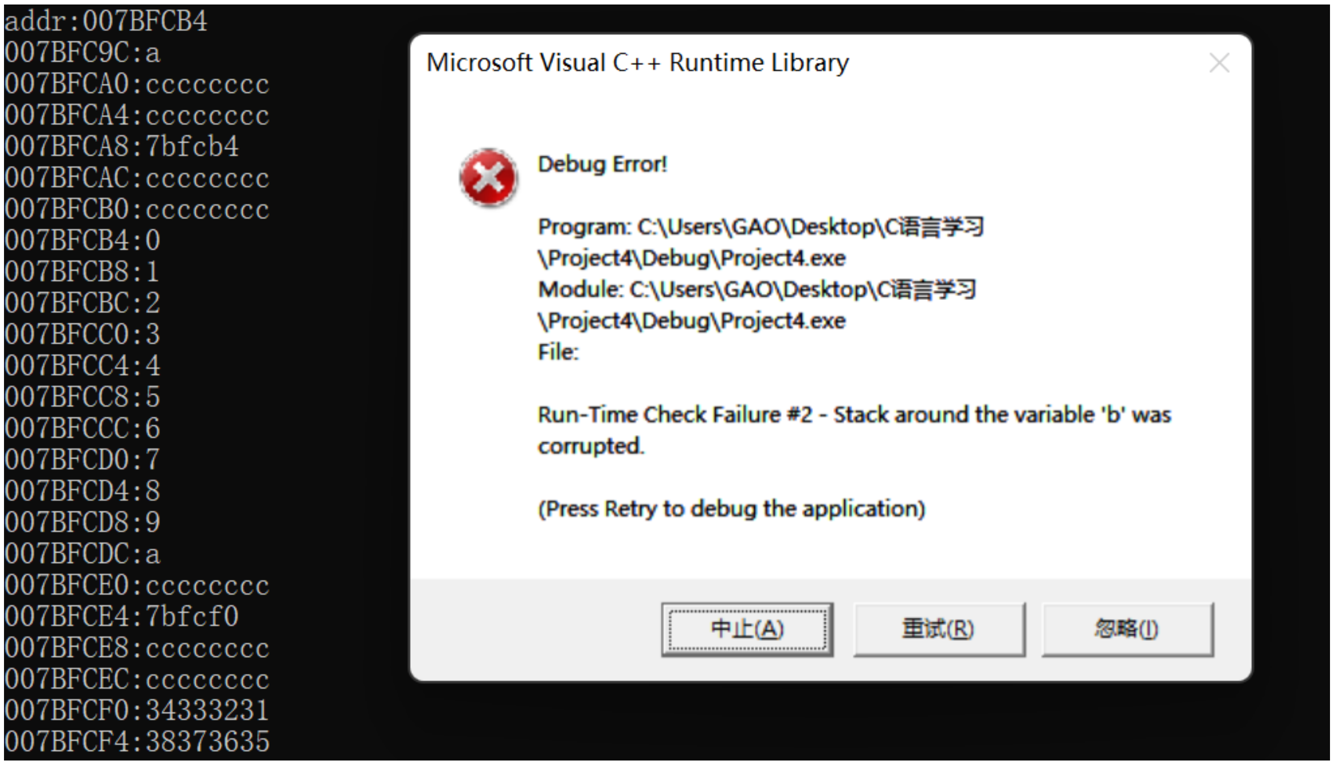
当我令 q[14]=14,q[15]=15 时，直接报错，这里规则有异



```
addr:0095FD14
0095FCFC:a
0095FD00:cccccccc
0095FD04:cccccccc
0095FD08:95fd14
0095FD0C:cccccccc
0095FD10:cccccccc
0095FD14:0
0095FD18:1
0095FD1C:2
0095FD20:3
0095FD24:4
0095FD28:5
0095FD2C:6
0095FD30:7
0095FD34:8
0095FD38:9
0095FD3C:cccccccc
0095FD40:cccccccc
0095FD44:95fd50
0095FD48:cccccccc
0095FD4C:e
0095FD50:f
0095FD54:38373635
0095FD58:cccc0039
```

```
Microsoft Visual C++ Runtime Library

Debug Error!

Program: C:\Users\GAO\Desktop\C语言学习
\Project4\Debug\Project4.exe
Module: C:\Users\GAO\Desktop\C语言学习
\Project4\Debug\Project4.exe
File:

Run-Time Check Failure #2 - Stack around the variable 'a' was
corrupted.

(Press Retry to debug the application)
```

当我再令 q[10]=10 时，也报错了



```
addr:007BFCB4
007BFC9C:a
007BFCA0:cccccccc
007BFCA4:cccccccc
007BFCA8:7bfcb4
007BFCAC:cccccccc
007BFCB0:cccccccc
007BFCB4:0
007BFCB8:1
007BFCBC:2
007BFCC0:3
007BFCC4:4
007BFCC8:5
007BFCCC:6
007BFCD0:7
007BFCD4:8
007BFCD8:9
007BFCDC:a
007BFCE0:cccccccc
007BFCE4:7bfcf0
007BFCE8:cccccccc
007BFCEC:cccccccc
007BFCF0:34333231
007BFCF4:38373635
```

Microsoft Visual C++ Runtime Library

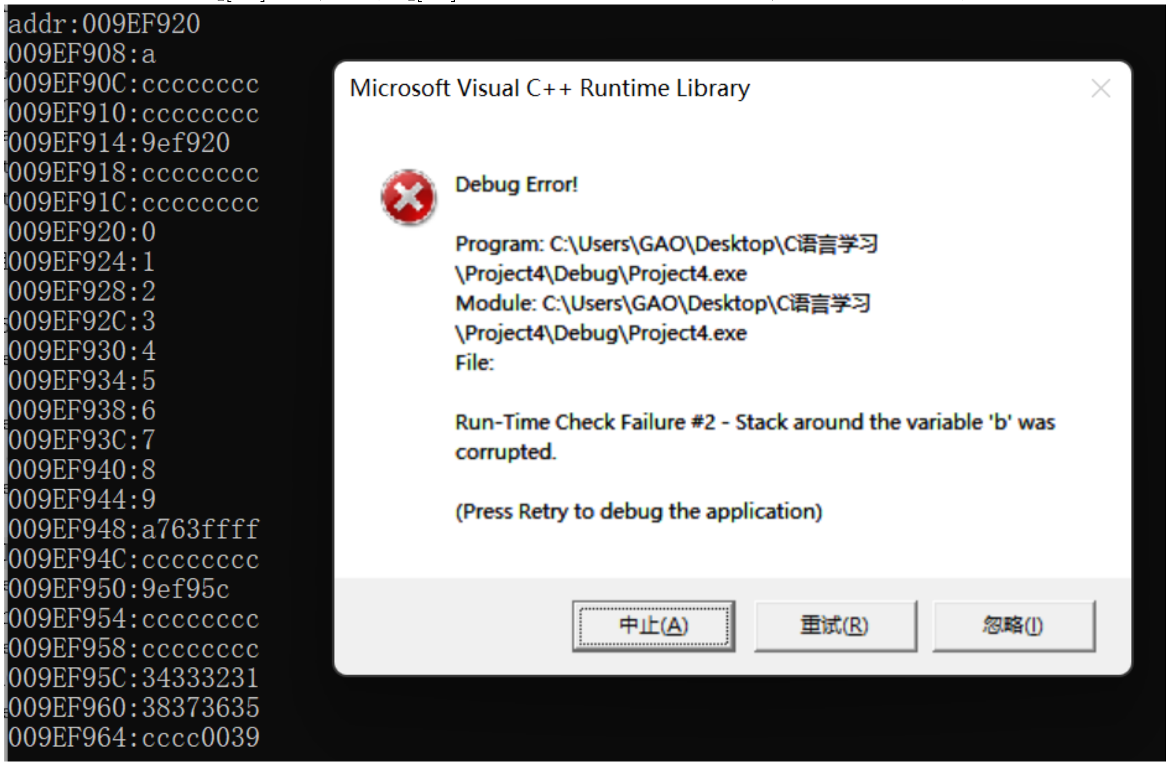
Debug Error!

Program: C:\Users\GAO\Desktop\C语言学习
\Project4\Debug\Project4.exe
Module: C:\Users\GAO\Desktop\C语言学习
\Project4\Debug\Project4.exe
File:

Run-Time Check Failure #2 - Stack around the variable 'b' was corrupted.

(Press Retry to debug the application)

但是当我注释掉 q[10]=10, 再令 q[10]=9999999999999999999 时, 报错了。这里规则也有异



```
addr:009EF920
009EF908:a
009EF90C:cccccccc
009EF910:cccccccc
009EF914:9ef920
009EF918:cccccccc
009EF91C:cccccccc
009EF920:0
009EF924:1
009EF928:2
009EF92C:3
009EF930:4
009EF934:5
009EF938:6
009EF93C:7
009EF940:8
009EF944:9
009EF948:a763ffff
009EF94C:cccccccc
009EF950:9ef95c
009EF954:cccccccc
009EF958:cccccccc
009EF95C:34333231
009EF960:38373635
009EF964:cccc0039
```

Microsoft Visual C++ Runtime Library

Debug Error!

Program: C:\Users\GAO\Desktop\C语言学习
\Project4\Debug\Project4.exe
Module: C:\Users\GAO\Desktop\C语言学习
\Project4\Debug\Project4.exe
File:

Run-Time Check Failure #2 - Stack around the variable 'b' was corrupted.

(Press Retry to debug the application)

总之，对于普通字符型数组识别越界的范围还是 4 个存储单元，但对于普通整型数组来说，只要越界就直接报错。

Part 3 Conclusion

本小节作业中，我使用了多种形式的测试程序来检测内存越界访问的可能性和后果。我在多个编译器环境下运行了相同的测试程序，发现不同的编译器对于内存越界访问的处理方式和报错信息有所差异，也观察了不同的测试程序对于内存越界访问的敏感程度，发现一些程序可以正常运行，而另一些程序会导致段错误或其他异常。通过这些实验，我对内存越界访问有了更深入的理解。我认识到内存越界访问是一种严重的编程错误，它可能会破坏程序的正确性和安全性，甚至影响其他进程或系统的稳定性。我也了解到操作系统和编译器都有一些机制来防范或检测内存越界访问，例如使用虚拟地址空间、保护页、栈保护、边界检查等。但是这些机制并不完美，有时候会漏掉一些潜在的问题或者给出不明确的提示。

因此，我应该养成良好的编程习惯来尽量防范内存越界访问。例如，在定义数组或指针时要注意分配合适的大小和类型，在使用数组或指针时要注意检查索引或地址是否合法，在释放数组或指针时要避免重复释放或野指针，在调用函数时要保证参数和返回值与函数声明一致，在写循环或条件语句时要避免死循环或逻辑错误等。此外，我还应该利用一些工具来辅助我的编程过程，例如使用调试器、静态分析器、动态分析器等来发现和修复我的代码中可能存在的内存越界访问问题。