

# C++面向对象程序设计 作业报告2

报告人：邓雍 学号：2022141220184

## 一.多文件编译的使用

CMake能打包多文件一同编译，实现不同cpp文件间的串联。下给出CMake编译多文件系统的简单过程书写。

首先分步编写解决问题所需的文件

```
C stuinfo.h ×
Assignment_2 > stuinfo > include > C stuinfo.h > stuinfo
1  #pragma once
2
3  struct stuinfo
4  {
5      char name[20];
6      double score[3];
7      double ave;
8  };
9  void inputstu(stuinfo stu[],int n);
10 void showstu(stuinfo stu[],int n);
11 void sortstu(stuinfo stu[],int n);
12 bool findstu(stuinfo stu[],int n,char ch[]);
```

首先编写的是所需的头文件，我们在其中定义了一个结构体和四个函数，在其他cpp内调用#include"stuinfo.h"便可调用该头文件内对应内容

stuinfo.cpp 2 X

Assignment\_2 > stuinfo > src > stuinfo.cpp > showstu(stuinfo [], int)

```
1  #include<iostream>
2  #include<cstdio>
3  #include<cstring>
4  #include<algorithm>
5  #include"stuinfo.h"
6  using namespace std;
7
8  void inputstu(stuinfo stu[],int n)
9  {
10     for(int i=0;i<n;i++)
11     {
12         scanf("%s",stu[i].name);
13         double sum=0;
14         for(int j=0;j<3;j++) scanf("%lf",&stu[i].score[j]),sum+=stu[i].score[j];
15         stu[i].ave=sum/3;
16     }
17 }
18
19 void showstu(stuinfo stu[],int n)
20 {
21     for(int i=0;i<n;i++)
22     {
23         cout<<stu[i].name<<" ";
24         for(int j=0;j<3;j++) printf("%.2lf ",stu[i].score[j]);
25         printf("%.2lf\n",stu[i].ave);
26     }
27 }
28
29 bool cmp(stuinfo a,stuinfo b)//定义结构体比较方法
30 {
31     return a.ave>b.ave;
32 }
33 void sortstu(stuinfo stu[],int n)
34 {
35     sort(stu,stu+n,cmp);//结构体快速排序,按平均分从大到小排序
36 }
37
38 bool check(char a[],char b[])//bool返回值为1代表a,b两char[]相同,为0代表不同
39 {
40     int len=strlen(a),len2=strlen(b);
41     if(len!=len2) return 0;
42     for(int i=0;i<len;i++) if(a[i]!=b[i]) return 0;
43     return 1;
44 }
45
46 bool findstu(stuinfo stu[],int n,char ch[])
47 {
48     for(int i=0;i<n;i++) if(check(stu[i].name,ch)) return 1;
49     return 0;
50 }
```

其次是根据函数功能需求实现相应四个函数，当头文件与该cpp一同编译时就可直接在其他cpp中调用函数实现功能。

main.cpp 2 X

Assignment\_2 > stuiinfo > src > main.cpp > main()

```
1  #include<iostream>
2  #include<cstdio>
3  #include"stuiinfo.h"
4
5  using namespace std;
6  const int MAXN=20000;
7  stuiinfo a[MAXN+5];
8  int n;
9  char s[20];
10 int main()
11 {
12     scanf("%d",&n);
13     inputstu(a,n);
14     showstu(a,n);
15     sortstu(a,n);
16     showstu(a,n);
17     scanf("%s",s);
18     if(findstu(a,n,s)) puts("Ok");
19     else puts("Not Found");
20 }
```

最后写一个简单的主函数来验证我们实现的四种功能，读入人数n，学生数据，输出数据排序再输出数据，查找特定学生

## M CMakeLists.txt X

Assignment\_2 > stuinfo > M CMakeLists.txt

```
1  #cmake_minimum_required(VERSION 3.18)
2
3  project(stuinfo)
4
5  aux_source_directory(./src DIR_SRCS)
6
7  include_directories(include)
8
9  add_executable(stuinfo ${DIR_SRCS})
```

分步编写完所需的文件后，我们写一个CMakeLists来完成我们的编译统筹，将cpp文件放入src文件夹，头文件放入include文件夹，整体编译与调用即实现了多文件编译

```
root@457ba46ceb43:/ws/Assignment_2/stuinfo# cmake CMakeLists.txt
CMake Warning (dev) in CMakeLists.txt:
  No cmake_minimum_required command is present.  A line of code such as

    cmake_minimum_required(VERSION 3.18)

  should be added at the top of the file.  The version specified may be lower
  if you wish to support older CMake versions for this project.  For more
  information run "cmake --help-policy CMP0000".
This warning is for project developers.  Use -Wno-dev to suppress it.

-- Configuring done
-- Generating done
-- Build files have been written to: /ws/Assignment_2/stuinfo
```

Cmake运行后得到相应的CMakeFiles文件，再度Make得到可执行文件

```
root@457ba46ceb43:/ws/Assignment_2/stuinfo# make
Scanning dependencies of target stuinfo
[ 33%] Building CXX object CMakeFiles/stuinfo.dir/src/stuinfo.o
[ 66%] Linking CXX executable stuinfo
[100%] Built target stuinfo
```

```
root@457ba46ceb43:/ws/Assignment_2/stuinfo# ./stuinfo
2
a 1 1 1
b 2 2 2
a 1.00 1.00 1.00 1.00
b 2.00 2.00 2.00 2.00
b 2.00 2.00 2.00 2.00
a 1.00 1.00 1.00 1.00
a
Ok
```

运行测试样例得到正确结果，附运行图

## 二.Types

---

### 1.static用法及作用

**static用法:**声明在变量前为变量添加额外属性

**static作用:**

- 1)隐藏: 未声明static的全局变量和函数本身自带全局可见性, 即不用额外再次声明
- 2)保持变量内容持久: 程序刚开始即完成唯一一次初始化, 存储在静态存储区
- 3)默认初始化为0: 静态存储区所有字节默认为0
- 4)类成员中声明: 作用于整个函数体, 只可被模块内函数使用, 属于整个类所有

### 2.隐式转换

**隐式转换是什么:**即自动类型转换, 即不需要代码书写, 系统自动完成的类型转换

**消除隐式转换的方法:**在构造函数声明的时候加上explicit关键字即可禁止隐式转换

### 3.程序解释

```
test1.cpp
Assignment_2 > Type > test1.cpp
1  #include <iostream>
2
3  using std::cout;
4  using std::endl;
5
6  int main() {
7      int num1 = 1234567890;
8      int num2 = 1234567890;
9      int sum = num1 + num2;
10     cout << "sum = " << sum << endl;
11
12     float f1 = 1234567890.0f;
13     float f2 = 1.0f;
14     float fsum = f1 + f2;
15     cout << "fsum = " << fsum << endl;
16     cout << "(fsum == f1) is " << (fsum == f1) << endl;
17     float f = 0.1f;
18     float sum10x = f + f + f + f + f + f + f + f + f + f;
19     float mul10x = f * 10;
20
21     cout<<"sum10x = " << sum10x << endl;
22     cout<<"mul10x = " << mul10x << endl;
23     cout<<"(sum10x == 1) is " << (sum10x == 1.0) << endl;
24     cout<<"(mul10x == 1) is " << (mul10x == 1.0) << endl;
25     return 0;
26 }
```

问题 输出 调试控制台 终端 端口

```
root@457ba46ceb43:/ws# g++ test1.cpp -o test1
cc1plus: fatal error: test1.cpp: No such file or directory
compilation terminated.
root@457ba46ceb43:/ws# cd Assignment_2
root@457ba46ceb43:/ws/Assignment_2# cd Type
root@457ba46ceb43:/ws/Assignment_2/Type# g++ test1.cpp -o test1
root@457ba46ceb43:/ws/Assignment_2/Type# ./test1
sum = -1825831516
fsum = 1.23457e+09
(fsum == f1) is 1
sum10x = 1
mul10x = 1
(sum10x == 1) is 0
(mul10x == 1) is 1
root@457ba46ceb43:/ws/Assignment_2/Type#
```

程序代码与运行截图如上，下给出程序运行具体解释

- 1)将两个int变量num1与num2相加并赋值给sum，然后输出sum，因为两个int值相加超过了int的范围，所以发生了溢出，sum溢出成为负数
- 2)两个float变量相加赋值给fsum，但因为float精度不足，所以fsum仍然与f1视为相等
- 3)0.1转化为二进制存储时不能被完全表示，存在误差，加法放大了这个误差，所以在与1比较时加法误差大于乘法误差。故加法输出0，乘法输出1

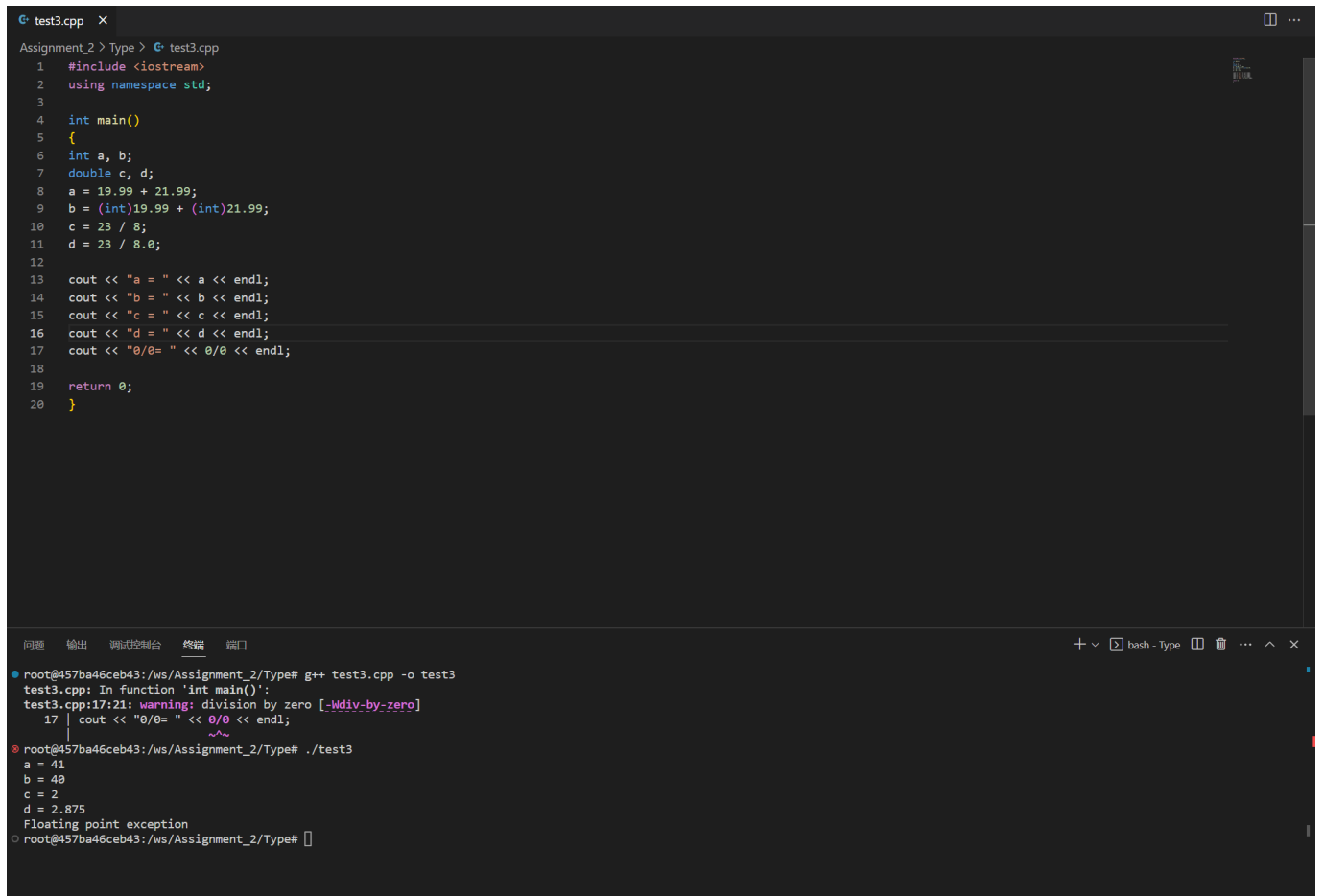
```
Assignment_2 > Type > test2.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6  cout << fixed;
7  float f1 = 1.0f;
8  cout<<"f1 = "<<f1<<endl;
9  float a = 0.1f;
10 float f2 = a+a+a+a+a+a+a+a;
11 cout<<"f2 = "<<f2<<endl;
12
13 if(f1 == f2)
14 cout << "f1 = f2" << endl;
15 else
16 cout << "f1 != f2" << endl;
17
18 return 0;
19 }
```

问题 输出 调试控制台 终端 窗口

```
root@457ba46ceb43:/ws/Assignment_2/Type# g++ test2.cpp -o test2
root@457ba46ceb43:/ws/Assignment_2/Type# ./test2
f1 = 1.000000
f2 = 1.000000
f1 != f2
root@457ba46ceb43:/ws/Assignment_2/Type#
```

## 程序代码与运行截图如上，下给出程序运行具体解释

f1,f2在输出精度范围内相同，但是由于0.1二进制存储精度问题,实际上a不等于0.1，在加法过程中这部分多余的值也会被算入，所以最终值会偏离0.1，所以即使在输出精度中相同的情况下，两者的实际值依然是不同的



The screenshot shows a C++ IDE with a file named `test3.cpp`. The code defines a `main` function that declares variables `a` (int), `b` (int), `c` (double), and `d` (double). It performs calculations: `a = 19.99 + 21.99`, `b = (int)19.99 + (int)21.99`, `c = 23 / 8`, and `d = 23 / 8.0`. It then prints the values of `a`, `b`, `c`, and `d`, followed by a division by zero: `cout << "0/0= " << 0/0 << endl`.

The output window shows the compilation and execution results. It indicates a warning for division by zero in the `cout` statement. The output shows the values of `a` (41), `b` (40), `c` (2), and `d` (2.875), followed by a floating point exception error.

程序代码与运行截图如上，下给出程序运行具体解释

- 1)在c++中double转int是直接舍去小数部分的内容，故现将两者相加舍去的会更少，而先转化再加会舍去更多，所以`a=41`，`b=40`
- 2)c++中整数的除法都是直接舍去小数部分的，即整数除法的结果是带余除法中的不完全商，而小数是正常的除法。c++中除法只要有一部分是实数就整体会做实数除法。故`c=2`，`d=2.875`
- 3)c++中不存在0除以0

## 四.Structs

### 1.结构体内存对齐问题



```

2 struct Info {
3     uint8_t a;
4     uint16_t b;
5     uint8_t c;

```

5

#### 4.2 Exercise

#### 4 STRUCTS

```

6 };
7 std::cout << sizeof(Info) << std::endl; // 6 2 + 2 + 2
8 std::cout << alignof(Info) << std::endl; // 2
9
10 struct alignas(4) Info2 {
11     uint8_t a;
12     uint16_t b;
13     uint8_t c;
14 };
15 std::cout << sizeof(Info2) << std::endl; // 8 4 + 4
16 std::cout << alignof(Info2) << std::endl; // 4

```

---

```

1 // alignas 失效的情况
2 struct Info {
3     uint8_t a;
4     uint32_t b;
5     uint8_t c;
6 };
7 std::cout << sizeof(Info) << std::endl; // 12 4 + 4 + 4
8 std::cout << alignof(Info) << std::endl; // 4
9 struct alignas(2) Info2 {
10     uint8_t a;
11     uint32_t b;
12     uint8_t c;
13 };
14 std::cout << sizeof(Info2) << std::endl; // 12 4 + 4 + 4
15 std::cout << alignof(Info2) << std::endl; // 4

```

---

观察上图中alignas生效与非生效情况：我们可以发现当结构体最大内存小于等于alignas传入参数时，可以成功对齐，当结构体最大内存时大于传入参数时，对齐失效。

## 2.Exercise

```

Assignment 2 > Struct > include > C exercise.h
1  #pragma once
2  #include<cmath>
3  using namespace std;
4
5  const double eps=1e-7;
6
7  struct Point{double x,y;};
8  using Vec =Point;
9  struct Line {Point P;Vec v;};
10 struct Seg{Point A,B;};
11 struct Circle(Point O,double r;);
12 const Point O={0,0};
13 const Line Ox={0,{1,0}},Oy={0,{1,0}};
14 const double PI=acos(-1),EPS=1e-9;
15
16 pair<bool, Point> barycenter(Point A, Point B, Point C);
17 pair<bool, Point> circumcenter(Point A, Point B, Point C);
18 pair<bool, Point> incenter(Point A, Point B, Point C);
19 pair<bool, Point> orthocenter(Point A, Point B, Point C);

```

新建一个项目同名头文件，定义我们需要的函数与结构体等变量

```

Assignment 2 > Struct > src > C exercise.cpp
1  #include<iostream>
2  #include<iomanip>
3  #include<string>
4  #include<cmath>
5  #include<queue>
6  #include<algorithm>
7  #include"exercise.h"
8  using namespace std;
9
10 Point operator -(Point A,Point B){return {A.x-B.x,A.y-B.y};}
11 double Cross(Point A,Point B)
12 {
13     return A.x*B.y-A.y*B.x;
14 }
15
16 double sqr(double x){return x*x;}
17
18 double dis(Point A,Point B)
19 {
20     return sqrt(sqr(A.x-B.x)+sqr(A.y-B.y));
21 }
22
23 bool check(Point A,Point B,Point C)
24 {
25     return fabs(Cross(B-A,C-A))<eps;
26 }
27
28 pair<bool, Point> barycenter(Point A, Point B, Point C)
29 {
30     if(!check(A,B,C)) return {0,0};
31     double X=(A.x+B.x+C.x)/3,Y=(A.y+B.y+C.y)/3;
32     return {X,Y};
33 }
34
35 pair<bool, Point> circumcenter(Point A, Point B, Point C)
36 {
37     if(!check(A,B,C)) return {0,0};
38     double A1,B1,C1,A2,B2,C2;
39     A1=2*(B.x-A.x);
40     B1=2*(B.y-A.y);
41     C1=sqr(sqr(B.x-A.x)+sqr(B.y-A.y))-sqr(A.x-A.x-A.y*A.y);
42     A2=2*(C.x-A.x);
43     B2=2*(C.y-A.y);
44     C2=sqr(sqr(C.x-A.x)+sqr(C.y-A.y))-sqr(A.x-A.x-A.y*A.y);
45     double X=(B1*C2-C1*B2)/(A1*B2-A2*B1);
46     double Y=(A1*C2-A2*C1)/(A1*B2-A2*B1);
47     return {X,Y};
48 }
49
50 pair<bool, Point> incenter(Point A, Point B, Point C)
51 {
52     double d1=dis(B,C);
53     double d2=dis(A,C);
54     double d3=dis(A,B);
55     double S=d1*d2*d3;
56     double X=(d1*A.x+d2*B.x+d3*C.x)/S;
57     double Y=(d1*A.y+d2*B.y+d3*C.y)/S;
58     return {X,Y};
59 }
60
61 pair<bool, Point> orthocenter(Point A, Point B, Point C)
62 {
63     double A1=C.x-B.x;
64     double B1=C.y-B.y;
65     double C1=0;
66
67     double A2=C.x-A.x;
68     double B2=C.y-A.y;
69     double C2=(B.x-A.x)*B2-(B.y-A.y)*B2;
70
71     double d=A1*B2-A2*B1;
72     double X=A.x+(C1*B2-C2*B1)/d;
73     double Y=A.y+(A1*C2-A2*C1)/d;
74     return {X,Y};
75 }

```

在同名cpp文件中实现定义求重心，内心，外心，垂心的四个函数

```

Assignment_2 > Struct > src > main.cpp
1  #include<iostream>
2  #include<iomanip>
3  #include<cstring>
4  #include<cstdio>
5  #include<queue>
6  #include<algorithm>
7  #include"exercise.h"
8  using namespace std;
9  Point A,B,C;
10 pair<bool, Point> S1,S2,S3,S4;
11 int main()
12 {
13     scanf("%lf%lf%lf%lf%lf",&A.x,&A.y,&B.x,&B.y,&C.x,&C.y);
14     S1=barycenter(A,B,C);
15     S2=circumcenter(A,B,C);
16     S3=incenter(A,B,C);
17     S4=orthocenter(A,B,C);
18     if(!S1.first) puts("Warning");
19     else
20     {
21         printf("%lf %lf\n",S1.second.x,S1.second.y);
22         printf("%lf %lf\n",S2.second.x,S2.second.y);
23         printf("%lf %lf\n",S3.second.x,S3.second.y);
24         printf("%lf %lf\n",S4.second.x,S4.second.y);
25     }
26 }

```

主函数调用四个函数检验基本功能实现的正确性

```

Assignment_2 > Struct > CMakeLists.txt
1  #cmake_minimum_required(VERSION 3.18)
2
3  project(Struct)
4
5  add_library(lib SHARED ./src/exercise.cpp)
6
7  add_executable(Struct ./src/main.cpp ./include/exercise.h)
8
9  target_link_libraries(Struct lib)
10
11 include_directories(include)

```

撰写文件编译所需配置文件CMakeLists，通过在CMakeLists文件中编写命令以创建共享库，将exercise.cpp文件中的函数编译到共享库中，并利用target\_link\_libraries指令使可执行文件连接到共享库中

```

root@457ba46ceb43:/ws/Assignment_2/Struct/build# cmake ..
CMake Warning (dev) in CMakeLists.txt:
  No cmake_minimum_required command is present.  A line of code such as

    cmake_minimum_required(VERSION 3.18)

  should be added at the top of the file.  The version specified may be lower
  if you wish to support older CMake versions for this project.  For more
  information run "cmake --help-policy CMP0000".
This warning is for project developers.  Use -Wno-dev to suppress it.

-- Configuring done
-- Generating done
-- Build files have been written to: /ws/Assignment_2/Struct/build
root@457ba46ceb43:/ws/Assignment_2/Struct/build# make
[ 50%] Built target lib
[100%] Built target Struct
root@457ba46ceb43:/ws/Assignment_2/Struct/build# ./Struct
0 0 10 0 10
3.333333 3.333333
5.000000 5.000000
2.928932 2.928932
0.000000 0.000000
root@457ba46ceb43:/ws/Assignment_2/Struct/build# ./Struct
2 2 3 4 5 2
3.333333 2.666667
3.500000 2.500000
3.203820 2.744002
3.000000 3.000000
root@457ba46ceb43:/ws/Assignment_2/Struct/build# ./Struct
0 0 1 0 2 0
Warning

```

编译并运行测试样例，测试样例正确，证明上述内容实现准确

## 五.C++动态内存申请

### 1.C++的内存分区

---

```
1 class A {
2     int num;
3 }
4
5 static int a;    //(1)
6 auto b=0;        //(2)
7
8 int main()
9 {
10     char s[]="abc";    //(3)
11     char *p="123456";  //(4)。
12
13     p1= (char *)malloc(10); // (5)
14     A *a = new A();      // (6)
15 }
```

---

(1)存储在全局/静态存储区

(2)存储在全局/静态存储区

(3)`S[]`存储在栈中, "abc"存储在常量存储区

(4)\*`p`存储在栈中, "123456"存储在常量存储区

(5)`malloc`分配的空间在堆中, `p1`仍在全局/静态存储区

(6)`new`分配的空间存储在堆中, \*`a`仍存储在栈中

## 2.问答题

(1)`new`和`malloc`区别:`new`是需要编译器支持的关键字运算符, 会自动调用构造函数计算类型大小, 返回类型化指针, 可重载与自定义内存分配策略。而`malloc`本身是一个函数, 不会再调用构造函数, 需要自己指定字节数, 返回`void`指针, 需强制类型转换, 不可重载, 不可自定义内存分配策略。

(2)`delete p` 用于释放一个指针 `p` 所指向的动态分配内存。区别在于`delete p`只能释放`new`所定义的内存, 且只会调用一次析构函数, 不会释放栈上的空间, 而`delete[]p`会调用每个成员的析构函数`allocator`可以分配内存不必使用`new`, 可用`allocate`函数分配内存, `deallocate`函数释放内存

(3)`malloc`不调用构造函数和析构函数, `delete`运算符会调用对象的析构函数并释放内存, 所以不能将`malloc`分配的内存用`delete`运算符直接释放。`malloc`函数需要使用`free`函数来释放

(4)`malloc`函数在程序运行期间在堆区申请一块指定大小的内存供程序使用, 内部会记录已经分配的内存块并按一定规则存放, 避免内存碎片化, 以及可以更快地释放内存。`free`函数会将对应的内存空间标记为已释放, 便于下次`malloc`调用时可以重用。

## 3.Exercise

Assignment\_2 > Memory > include > C RandomGenerator.h

```
1  #pragma once
2
3  int* create(int n);
4  void Print(int* A,int n);
5  void Delete(int* A);
```

Assignment\_2 > Memory > src > C RandomGenerator.cpp

```
1  #include"RandomGenerator.h"
2  #include<iomanip>
3  #include<iostream>
4  #include<cstring>
5  #include<cstdio>
6  #include<ctime>
7  #include<stdlib.h>
8
9  using namespace std;
10
11 int* create(int n)
12 {
13     int* A=new int[n];
14     srand(time(NULL));
15     for(int i=0;i<n;i++) A[i]=rand()%n;
16     return A;
17 }
18
19 void Print(int* A,int n)
20 {
21     int maxx=0,minn=n-1;
22     for(int i=0;i<n;i++) maxx=max(maxx,A[i]),minn=min(minn,A[i]);
23     printf("%d %d\n",minn,maxx);
24 }
25
26 void Delete(int* A)
27 {
28     delete []A;
29 }
```

Assignment\_2 > Memory > src > C main.cpp

```
1  #include"RandomGenerator.h"
2  #include<iomanip>
3  #include<iostream>
4  #include<cstring>
5  #include<cstdio>
6  #include<ctime>
7
8  using namespace std;
9
10 int main()
11 {
12     int n;scanf("%d",&n);
13     int* A=create(n);
14     Print(A,n);
15     Delete(A);
16 }
```

Assignment\_2 > Memory > build > CMakeLists.txt

```
1  #cmake_minimum_required(VERSION 3.18)
2
3  project(Memory)
4
5  aux_source_directory(..src DIR_SRCS)
6
7  include_directories(..include)
8
9  add_executable(Memory ${DIR_SRCS})
```

在RandomGenerator.h文件中实现了函数命名，RandomGenerator.cpp文件中实现了函数，main.cpp文件中实现了整体函数的调用，CMakeLists.txt文件实现了文件编译。

```

root@457ba46ceb43:/ws/Assignment_2/Memory/build# cmake CMakeLists.txt
CMake Warning (dev) in CMakeLists.txt:
  No cmake_minimum_required command is present.  A line of code such as

    cmake_minimum_required(VERSION 3.18)

  should be added at the top of the file.  The version specified may be lower
  if you wish to support older CMake versions for this project.  For more
  information run "cmake --help-policy CMP0000".
This warning is for project developers.  Use -Wno-dev to suppress it.

-- Configuring done
-- Generating done
-- Build files have been written to: /ws/Assignment_2/Memory/build
root@457ba46ceb43:/ws/Assignment_2/Memory/build# make
Scanning dependencies of target Memory
[ 33%] Building CXX object CMakeFiles/Memory.dir/ws/Assignment_2/Memory/src/main.o
[ 66%] Linking CXX executable Memory
[100%] Built target Memory
root@457ba46ceb43:/ws/Assignment_2/Memory/build# ./Memory
10
1 9
root@457ba46ceb43:/ws/Assignment_2/Memory/build# ./Memory
10
0 9
root@457ba46ceb43:/ws/Assignment_2/Memory/build# ./Memory
10
2 9
root@457ba46ceb43:/ws/Assignment_2/Memory/build#

```

测试发现相同的输入跑出的结果是不同的，证明了代码的随机性是存在的。

## 六.Debug和Release

### 2.如何判断动态申请越界(C方式)

malloc使用链表维护堆，申请空间前后位置不可更改，所以在VSdebug会报错。在VSdebug中空间是0xffffffff，linux下是0。内存越界是申请了不属于分配的内存，malloc实现的情况下，代码常常能进入未知的领域，但仍能运行

### 3.如何判断动态申请越界(C++方式)

对比C状态下，发现C++中没有报错了。new创建的对象访问越界会导致程序的直接崩溃，而malloc所创建的内存块，访问越界会访问其他错误位置，但仍然使程序正常运行，此时会出现诡异的输出数据。在C++中，分配内存调用会有异常反馈机制，而malloc不存在此机制，导致了现象的发生

### 4.如何判断普通数组的越界访问(C++方式)

```

Assignment_2 > Debug&release > Acpp > main()
1  #include<iostream>
2  using namespace std;
3  int a[10];char b[10];
4  int main()
5  {
6      a[11]=1,b[11]='a';
7  }

```

```

root@457ba46ceb43:/ws/Assignment_2/Debug&release# g++ -fsanitize=address -g A.cpp -o test
root@457ba46ceb43:/ws/Assignment_2/Debug&release# ./test
=====
==1803==ERROR: AddressSanitizer: global-buffer-overflow on address 0x00000040422c at pc 0x000000401202 bp 0x7ffd7bf17290 sp 0x7ffd7bf17288
WRITE of size 4 at 0x00000040422c thread T0
#0 0x401201 in main /ws/Assignment_2/Debug&release/A.cpp:6
#1 0x7fa2f835d09 in __libc_start_main ../csu/libc-start.c:308
#2 0x401109 in _start (/ws/Assignment_2/Debug&release/test+0x401109)

0x00000040422c is located 4 bytes to the right of global variable 'a' defined in 'A.cpp:3:5' (0x404200) of size 40
0x00000040422c is located 52 bytes to the left of global variable 'b' defined in 'A.cpp:3:16' (0x404260) of size 10
SUMMARY: AddressSanitizer: global-buffer-overflow /ws/Assignment_2/Debug&release/A.cpp:6 in main
Shadow bytes around the buggy address:
 0x000000787f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x00000078800: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x00000078810: 00 00 00 00 00 00 00 00 00 00 00 00 f9 f9 f9
 0x00000078820: f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9
 0x00000078830: f9 f9 f9 f9 00 00 01 f9 f9 f9 f9 f9 f9 f9 f9
->0x00000078840: 00 00 00 00[f9]f9 f9 f9 f9 f9 02 f9 f9
 0x00000078850: f9 f9 f9 f9 00 00 00 00 00 00 00 00 00 00 00
 0x00000078860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x00000078870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x00000078880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x00000078890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
Stack left redzone: f1
Stack mid redzone: f2
Stack right redzone: f3
Stack after return: f5
Stack use after scope: f8
Global redzone: f9
Global init order: f6
Poisoned by user: f7
Container overflow: fc
Array cookie: ac
Intra object redzone: bb
ASan internal: fe
Left alloca redzone: ca
Right alloca redzone: cb
Shadow gap: cc
==1803==ABORTING

```

用 `g++ -fsanitize=address -g A.cpp -o test` 命令得到可执行文件，运行可直接发现图中越界信息反馈

```

Assignment_2 > Debug&release > G: B.cpp > main()
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      int* a=new int[10];
6      char* b=new char [10];
7      a[11]=1,b[11]='a';
8  }

```

```

root@457ba46ceb43:/ws/Assignment_2/Debug&release# g++ -fsanitize=address -g B.cpp -o test2
root@457ba46ceb43:/ws/Assignment_2/Debug&release# ./test2
=====
==20695==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60400000003c at pc 0x000000401239 bp 0x7fff93ad7440 sp 0x7fff93ad7438
WRITE of size 4 at 0x60400000003c thread T0
#0 0x401238 in main /ws/Assignment_2/Debug&release/B.cpp:7
#1 0x7f684a0b8d09 in __libc_start_main ../csu/libc-start.c:308
#2 0x401119 in _start (/ws/Assignment_2/Debug&release/test2+0x401119)

0x60400000003c is located 4 bytes to the right of 40-byte region [0x604000000010,0x604000000038)
allocated by thread T0 here:
#0 0x7f684a6900b7 in operator new[](unsigned long) (/usr/local/lib64/libasan.so.6+0xb30b7)
#1 0x4011e7 in main /ws/Assignment_2/Debug&release/B.cpp:5
#2 0x7f684a0b8d09 in __libc_start_main ../csu/libc-start.c:308

SUMMARY: AddressSanitizer: heap-buffer-overflow /ws/Assignment_2/Debug&release/B.cpp:7 in main
Shadow bytes around the buggy address:
 0x0c087fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0c087fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0c087fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0c087fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0c087fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
->0x0c087fff8000: fa fa 00 00 00 00[fa]fa fa fa fa fa fa fa
 0x0c087fff8010: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x0c087fff8020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x0c087fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x0c087fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x0c087fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
Stack left redzone: f1
Stack mid redzone: f2
Stack right redzone: f3
Stack after return: f5
Stack use after scope: f8
Global redzone: f9
Global init order: f6
Poisoned by user: f7
Container overflow: fc
Array cookie: ac
Intra object redzone: bb
ASan internal: fe
Left alloca redzone: ca
Right alloca redzone: cb
Shadow gap: cc
==20695==ABORTING

```

相似地我们得到动态申请的越界信息反馈，可发现与静态数组相同

在其他环境下结果相同

## 5.总结

很多情况下编辑器在越界时不会使程序CE或RE，程序仍能正常运行，只是由于错误的地址访问，有时会取出极端错误结果。内存越界有时编辑器会自动解释适应，此时我们需要提高警惕，在奇怪数据出现时思考是否是内存越界的问题。

平时使用数组时，可保持稍微多申请一小部分空间的习惯，例如N+5的空间申请，可以防止部分用到N+1却忘了申请的情况。

## 七.计算器

```
root@457ba46ceb43:/ws/Assignment_2/Calc/build# cmake CMakeLists.txt
CMake Warning (dev) in CMakeLists.txt:
  No cmake_minimum_required command is present.  A line of code such as

    cmake_minimum_required(VERSION 3.18)

  should be added at the top of the file.  The version specified may be lower
  if you wish to support older CMake versions for this project.  For more
  information run "cmake --help-policy CMP0000".
This warning is for project developers.  Use -Wno-dev to suppress it.

-- Configuring done
-- Generating done
-- Build files have been written to: /ws/Assignment_2/Calc/build
root@457ba46ceb43:/ws/Assignment_2/Calc/build# make
[100%] Built target Calc
root@457ba46ceb43:/ws/Assignment_2/Calc/build# ./Calc
2+3
5
5+2*3
11
(5+2)*3
21
((5+2*0)+6)*2
22
4/0
#INFINITE
sqrt(3.0)
1.73205
exp(3)
20.0855
abs(-300)
300
max(-1,3)
3
random()
1.62371e+09
sin(0.65)
0.605186
```

计算器运行结果如上。整体思路是考虑用两个栈模拟实现，一个栈存放运算符，一个栈存放运算数据。为运算符分别设定优先级，一个运算符进站前先弹出高优先级运算，+，-优先级设为1，\*，/优先级为2，函数优先级为3，左括号优先级为0，在右括号找到的位置弹栈寻找左括号。一个值得注意的点是根据上一个字符是否是(或不存在字符来判断是负号还是减号

代码过长，放在附件中了。