

buglab实验报告

罗思佳2021201679

1.shuffle

第一处错误

是swap函数里，应该加上 `if(a == b) return;` 因为main函数中，如果输入的要交换的数的下标相同 (`a == b`)，此时swap函数中是实参是两个地址相同的数，即 `swap(data[a], data[a])`，这样函数调用之后data[a]和data[b]都变为0了，因此需要加一个if语句。

由于此题的最终正确提交未能显示在排行榜中，这里附上最终提交记录链接：[提交记录 #14771 - RUC ICS 2021 \(men.ci\)](https://men.ci/jcs2021/#14771-RUC)

第二处错误

是n和m弄混，n指的是交换次数，m指的是数列元素个数，在 `if(a < 0 || a >= n || b < 0 || b >= n)` 中，应该把n改成m。

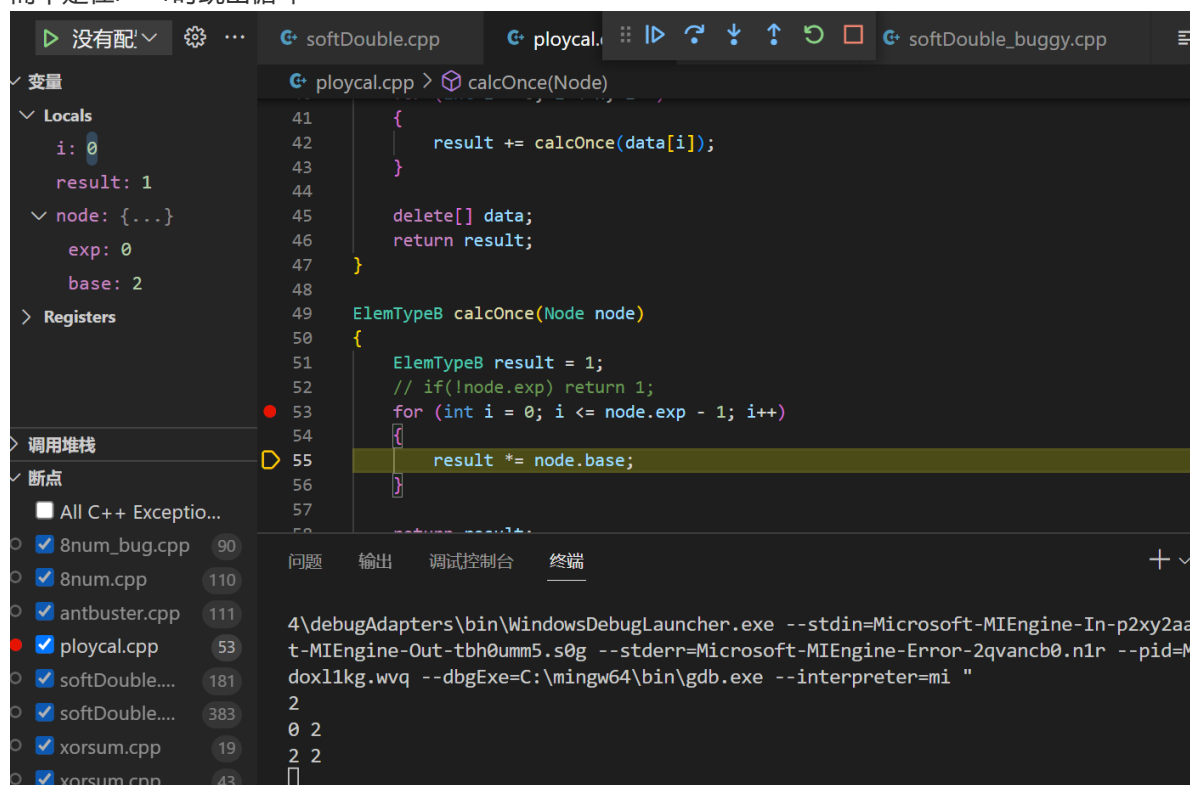
2.polycalc

第一处错误

`ElemTypeB result` 没有初始化，应该改为 `ElemTypeB result = 1;`，否则会默认是0。

第二处错误

当我输入“2 0 2 1 2”这一测试数据时，程序一直没有结束，电脑开始发热，调试时发现进入了死循环，而不是在 `i > -1` 时跳出循环



因此该代码没有考虑exp为0的情况，应该在for循环前加一个if判断语句: `if(!node.exp) return 1;`

3.violetStore

第一处错误

在类的构造函数中，items和prices开辟的是数组空间，所以在析构函数中要用 `delete []`。

第二处错误

由于price类中有string对象成员，在main函数中创建price对象时，用malloc分配内存不会调用string的构造函数，因而出错，需要改为new，对应的free要改为delete。

第三处错误

add_item() 函数中， `*items++ = name;` 和

`*prices++ = price;` 两个语句有错，因为指针++后就移动到了存数据的位置之后了，之后调用打印函数的时候就不能取到之前存的数据了，因此可以改成：

```
items[n] = name;
prices[n] = price;
n++;
```

或者

```
*(items+n) = name;
*(prices+n) = price;
n++;
```

4.swapCase

第一处错误

`scanf("%s", s);` 遇到空格会结束输入，因此改成 `scanf("%[^\n]", s);`，这种方式可以读入空格。

第二处错误

因为交上去总是显示超时，检查了一下发现for循环中每次都调用了strlen() 函数，会使时间成本升高导致超时，因此可以先设一个int变量，`int len = strlen(s)`，在for循环中用len替换strlen(s)。

5.xorsum

一处错误：Replace函数参数列表中的value和pos要换位置，因为Replace函数调用语句中是先调用后一个ReadInt函数再调用前一个ReadInt函数。

这里附上找错过程：

当输入题目中的样例时，第三行是“4 2”，因此Replace函数调用时理应是pos为4，value为2，而在调试过程中，我发现pos和value传反了，因此我将ReadInt函数复制了一份并改名为readInt，将Replace函数调用的那一行语句改为下图的样子：

```

for(int i=0;i<q;i++)
    Replace(ReadInt(), readInt());
printf("%d\n", ans);
return 0;

```

然后进入debug模式，发现输入“4 2”时，先调用了readInt函数而不是左边的ReadInt（如图），因此导致参数传反了。

```

8   int ReadInt(){
9       int x = 0;
10      char ch = getchar();
11      while(!isdigit(ch))
12          ch = getchar();
13      while(isdigit(ch)){
14          x = x * 10 + ch - '0';
15          ch = getchar();
16      }
17      return x;
18  }
19  int readInt(){
20      int x = 0;
21      char ch = getchar();
22      while(!isdigit(ch))
23          ch = getchar();
24      while(isdigit(ch)){
25          x = x * 10 + ch - '0';
26          ch = getchar();

```

之后上网查资料才发现，“C函数参数作为一个整体执行的顺序是从右向左,所以会先处理最右端的参数,然后依次向左处理。”

另一种方法是把ReadInt函数写在外面，不嵌套。

6.mergeIntervals

第一处错误

之前写程设二大作业的时候需要自己实现排序的cmp函数，所以有了解过compare函数需要遵循“严格弱序”的规则，因此这里要去掉等号。

第二处错误

问题出在 `last.r = it->r;`，区间合并时，应该要取 `last.r` 和 `it->r` 中的较大者，而不能简单的将 `it->r` 赋给 `last->r`。

7.8num

第一处错误

`State* curState = &curs;`，这一语句有问题，如果直接取curs地址的话，指针curState就直接指到了curs的地址。因为curs是while循环中的局部变量，每次循环都经历一次申请、释放，而局部变量是存放在栈中的，当一次while循环结束后，curs被释放，栈顶指针向高地址移动一个存储单元，下一次循环开始时，curs创建，此时curs的地址还是上一次的地址 没有变化，这样导致的后果是，之后产生的子结点的parent都指向了同一个地址，无法打印出完整的路径。

这里附上当时调试过程：可以看到打印的curs的地址都是同一个值

```
81     while(!que.empty()){
82         //请实现带深度限制的搜索
83         //提示：1) 状态比较 matrix2string(curState->matrix)==matrix2st
84         //      2) 移动空格位置 nextState->zero_x = curState->zero_x+r
85         //      3) 判断空格移动是否在允许范围 (nextState->zero_x<0||next
86         //      4) 防止状态重复
87         State curs = State(que.top().first);
88         State* curState = &curs;
89         std::cout<<"curState: "<<curState<<std::endl;
90         curDepth = que.top().second;
91         que.pop();
92         node_num--;
93         if(matrix2string(curState->matrix)==matrix2string(goal.matri
94         {
```

问题	输出	调试控制台	终端
	steps: curState: 0x63fca0		
	curState: 0x63fca0		
	curState: 0x63fca0		
	curState: 0x63fca0		
	curState: 0x63fca0		

行 90,

改法：不直接指向curs，对成员进行一一赋值，较为繁琐

```
State curs = State(que.top().first);
State* curState = (State*)malloc(sizeof(State));
curState->g = curs.g;
curState->parent = curs.parent;
curState->zero_x = curs.zero_x;
curState->zero_y = curs.zero_y;
memcpy(curState->matrix, curs.matrix, 9*sizeof(char));
curDepth = que.top().second;
que.pop();
node_num--;
```

优化后的改法：

不设curs，直接写 `State* curState = new State(que.top().first);`，即手动申请一个空间，避免了上述问题。对应的free要改成new。

第二处错误

错误出在判断是否找到目标状态的语句中的while语句，即如下两行：

```
curState = curState->parent;
free(curState);
```

作者本意是想将每次入栈后的curState释放，但 `curState = curState->parent` 会让curState指到curState->parent的地址，即这一地址有两个指针同时指到，这种情况直接 `free(curState)` 就会使curState->parent指向的内存被释放，从而引发悬空指针错误。（相关知识参考自<http://t.csdn.cn/x8OiH>）

修改：设一个临时指针temp指向curState的地址，curState移动后再delete temp。

第三处错误

由于nextState是用malloc手动申请的，因此最后需要用free进行手动释放。

8.segtree

识别出bug的过程

最初我以为是算法逻辑有错误，于是检查了很久、手动输入了很多数据测试，结果都没出错，而在平台上提交却通不过，回想之前做的几道题的错误，于是我开始思考是不是数据类型出错等“比较ics”的错误，于是我又去仔细阅读题干，发现

如果取极端情况，比如当n为10的5次方，ai为10的9次方时，数列的所有数加起来的和就是10的14次方，超过了int的表示范围，因此有些测试点会出现数据溢出的情况。

出错原因分析

对于一些数据比较大的情况，调用Query函数求和时，可能会超出int的表示范围，从而造成数据溢出。

修改解释

因为相加求和时可能超出int表示范围，所以将结构体中的sum改为long long型；同样的，将Query函数的返回值改为long long型，printf函数里的%d也要改为%lld。

ps：有一个地方后来复盘的时候发现改多余了，就是data[MAXN]的数据类型其实用int就够了，不用改成long long，因为当时没有仔细数INT_MAX的位数，以为10的9次方超范围了，也为了保险，所以没多想，就将data的数据类型一同改为了long long型。

9.antbuster

第一处错误

在函数DecreaseSign中，不能直接对所有的 `sign[i][j]` 进行减一，因为信息素最少为0，所以要加上一个判断条件，`if(sign[i][j] > 0)`。

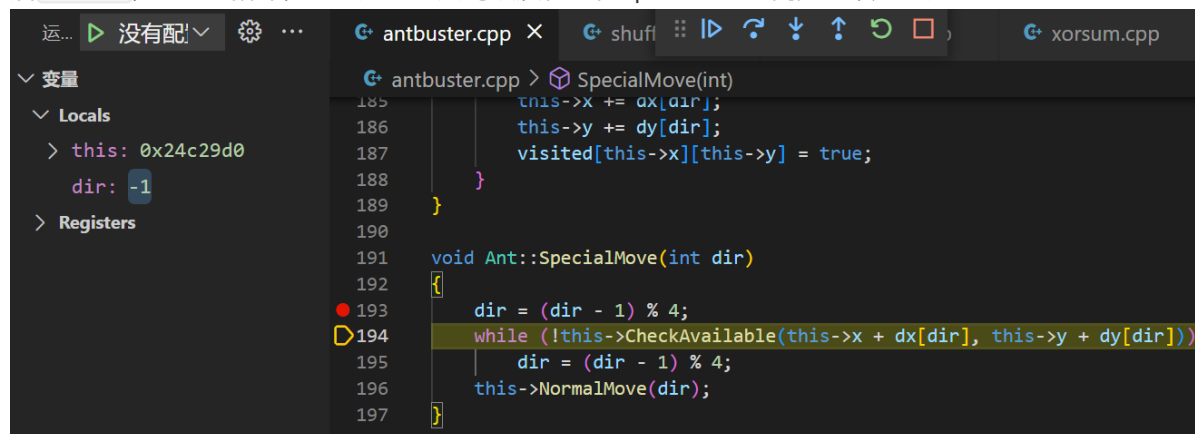
第二处错误

找出bug的过程和分析

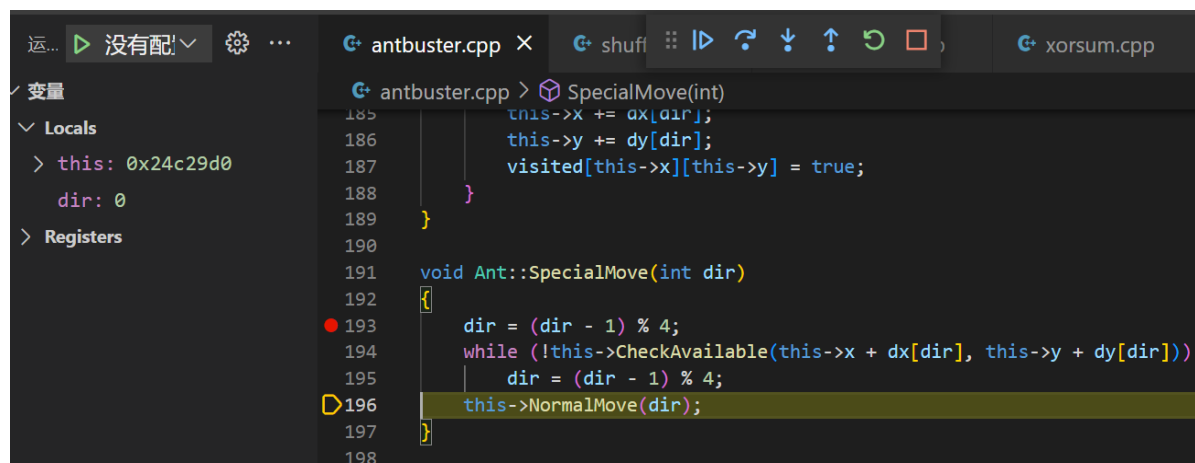
我先将样例的输入粘贴输进去，发现输出结果和样例输出相比，只有最早出生的蚂蚁的坐标不一样，应该是 (1,4) 而不是 (0,5)，即这只蚂蚁在第五秒时没有进行SpecialMove，而是NormalMove，初步怀疑是SpecialMove函数有bug，

```
vscode_c\homework\ICS_1\"antbuster
3 5
1 1 2
2 2
5
The game is going on
5
5 1 3 0 5
4 1 3 0 4
3 1 3 0 3
2 1 3 0 2
1 1 4 0 1
```

调试过程中很快发现，第五秒时dir变成-1了，然后在下面的while循环中一直循环，直到dir== -4，模4后dir==0，出while循环，才造成了这只蚂蚁没有进行SpecialMove而是继续向东走。



```
antbuster.cpp > SpecialMove(int)
185     this->x += dx[dir];
186     this->y += dy[dir];
187     visited[this->x][this->y] = true;
188 }
189
190
191 void Ant::SpecialMove(int dir)
192 {
193     dir = (dir - 1) % 4;
194     while (!this->CheckAvailable(this->x + dx[dir], this->y + dy[dir]))
195         dir = (dir - 1) % 4;
196     this->NormalMove(dir);
197 }
198
```



```
antbuster.cpp > SpecialMove(int)
185     this->x += dx[dir];
186     this->y += dy[dir];
187     visited[this->x][this->y] = true;
188 }
189
190
191 void Ant::SpecialMove(int dir)
192 {
193     dir = (dir - 1) % 4;
194     while (!this->CheckAvailable(this->x + dx[dir], this->y + dy[dir]))
195         dir = (dir - 1) % 4;
196     this->NormalMove(dir);
197 }
198
```

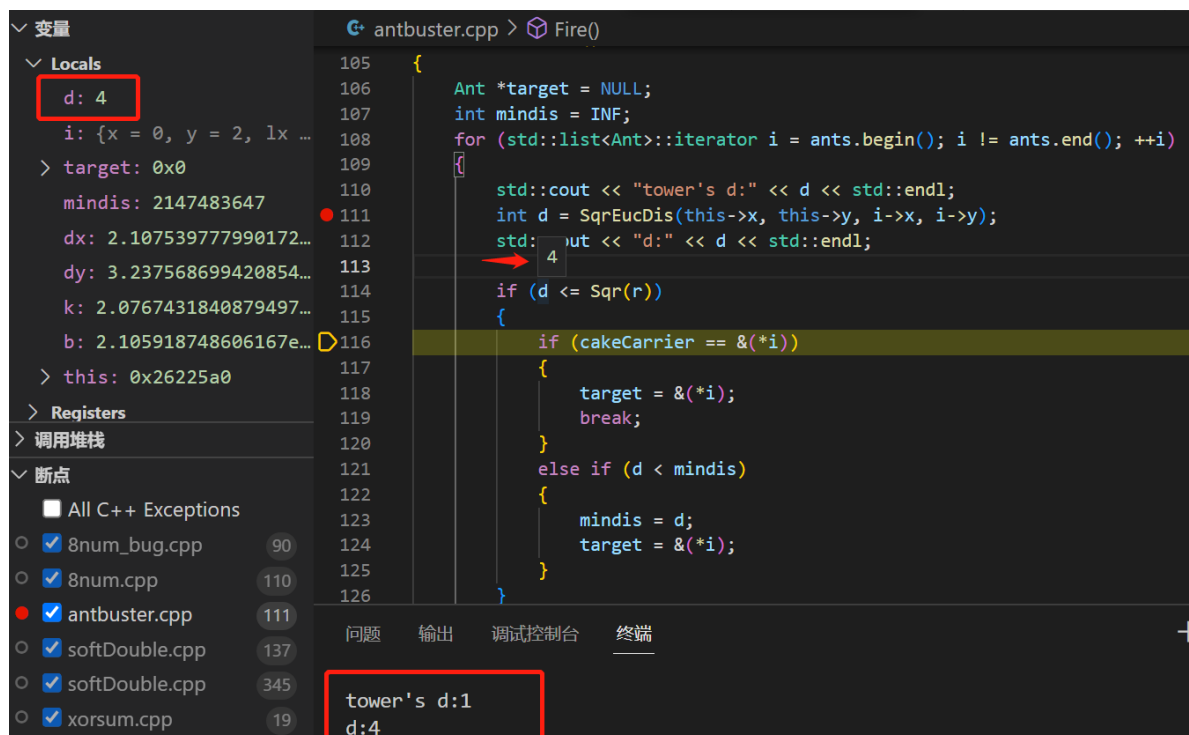
修改解释

在SpecialMove函数中，两句 `dir=(dir-1)%4`，`dir`可能会变成负数，导致走错方向，因此可以把-1改成+3。

第三处（不算错误的）错误

Tower类的Fire函数中，`int d=SqrEucDis(this->x,this->y,i->x,i->y);`，这里的局部变量`d`和全局变量`d`（Tower的单次攻击伤害）的名字重复了，虽然这里不改也能通过，但个人感觉区分一下会更好一点，这样不容易混淆，算是一种代码规范。

下图是测试局部变量是否能屏蔽全局变量的过程：在语句 `int d = SqrEucDis(this->x, this->y, i->x, i->y);`前，打印`d`的值是1，即单次攻击伤害；这条语句执行后，打印`d`的值为4，变成了塔和蚂蚁的欧拉距离，下面的if语句中的`d`的值依然是4，因此可以看出，同名的局部变量确实屏蔽掉了全局变量。



```
105 {
106     Ant *target = NULL;
107     int mindis = INF;
108     for (std::list<Ant>::iterator i = ants.begin(); i != ants.end(); ++i)
109     {
110         std::cout << "tower's d:" << d << std::endl;
111         int d = SqrEucDis(this->x, this->y, i->x, i->y);
112         std::cout << "d:" << d << std::endl;
113         if (d <= Sqr(r))
114         {
115             if (cakeCarrier == &(*i))
116             {
117                 target = &(*i);
118                 break;
119             }
120             else if (d < mindis)
121             {
122                 mindis = d;
123                 target = &(*i);
124             }
125         }
126     }
127 }
```

进一步查阅资料，弥补一下知识盲区：

1、局部变量能否和全局变量重名？

答：能，局部会屏蔽全局。要用全局变量，需要使用 `::`。

局部变量可以与全局变量同名，在函数内引用这个变量时，会用到同名的局部变量，而不会用到全局变量。对于有些编译器而言，在同一个函数内可以定义多个同名的局部变量，比如在两个循环体内都定义一个同名的局部变量，而那个局部变量的作用域就在那个循环体内。

第四处错误

根据题目对激光塔的描述，它在选定了打击目标后，只要目标在其射程内，塔到目标蚂蚁圆心的连线上的所有蚂蚁（这里“被打到”的判定变成了表示激光的线段与表示蚂蚁的圆有公共点）都会被打到并损`d`格血，所以我认为在SqrEucDis函数中，小于号应该改成小于等于，即 `Sqr(a) + Sqr(b) >= Sqr(a * x + b * y + c)` * 4，等号对应的情况是蚂蚁的圆刚好与塔和目标的连线相切，此时点到直线的距离刚好是0.5。

第五处错误

·找出bug的过程和出错原因分析

分析一下炮塔的Fire函数的逻辑，第一个for循环是在找离塔最近的蚂蚁或者背着蛋糕的蚂蚁，如果发现有背蛋糕的蚂蚁，就继续，计算塔和target的连线的斜率、截距，然后进入第二个for循环，这个循环是在找可以被激光“波及”的蚂蚁（包括target），即表示蚂蚁的圆和激光线段有公共点，for循环的if语句中，`Cross(k,-1.0,b,i->x,i->y)` 判断表示该蚂蚁的圆是否和激光直线有公共点，`InSegment(this->x,this->y,target->x,target->y,i->x,i->y)` 判断该蚂蚁是否在以塔和target连线为对角线的正方形内，从而将直线限定了范围，变成线段，第三个条件 `SqrEucDis(this->x, this->y, i->x, i->y) <= SqrEucDis(this->x, this->y, target->x, target->y)` 判断塔和该蚂蚁的距离是否小于塔和target的距离，只有三个条件都为真才执行“减血”操作。

因为之前觉得这里的逻辑比较复杂，所以反复分析了几遍，在看到第二个for循环之前的计算斜率和截距的语句段时，我突然想起在高中数学的解析几何常犯的错误，就是计算斜率时没有讨论斜率是否存在，以前在这里栽过好几次，因此印象比较深刻，所以发现这里的代码逻辑有问题，当塔、target、其他蚂蚁的x坐标都相同时，直线斜率不存在，计算出来 $k = \text{inf}$ ，代入到Cross函数中，本应该返回true，却因为 $k = \text{inf}$ 返回false，因此这里需要加上k不存在的情况。

·修改解释

需要加上塔、target、路过蚂蚁的x坐标相同的情况，即 `this->x == target->x && target->x == i->x`，然后与Cross函数进行“||”操作，用括号括起来。

10.softDouble

第一处错误

首先运行了一下，根据报错可知，1默认的是int型变量，因此左移的位数不能超过31位，因此需要在所有左移超过31位的1后面加上ULL的后缀，表示是unsigned long long型。

```
softDouble_buggy.cpp: In function 'bool isNaN(uint64_t)':
softDouble_buggy.cpp:143:63: warning: left shift count >= width of type [-Wshift-count-overflow]
    return (Exp(x) == (1 << 11) - 1) && (Fraction(x) & ((1 << 52) - 1)) != 0;
                                                                ^~

softDouble_buggy.cpp: In function 'bool isINF(uint64_t)':
softDouble_buggy.cpp:148:63: warning: left shift count >= width of type [-Wshift-count-overflow]
    return (Exp(x) == (1 << 11) - 1) && (Fraction(x) & ((1 << 52) - 1)) == 0;
                                                                ^~

softDouble_buggy.cpp: In function 'bool isZero(uint64_t)':
softDouble_buggy.cpp:153:24: warning: left shift count >= width of type [-Wshift-count-overflow]
    return (x & ((1 << 63) - 1)) == 0;
                       ^
```

第二处错误

找出bug的过程和原因分析

在阅读buggy代码中，我看到一个函数 `_mm_cvtsi64_si128(x)`，出现在 `write_to_string` 函数中，因为是第一次见，不知道它的作用是什么，结合上下文和题目要求 猜测是用来将64位的二进制串转换为64位双精度浮点数的，于是我上网查了一下这个函数，

Click or press 'S' to search, '?' for more options...

Function core::arch::x86_64::_mm_cvtsi64_si128

```
pub unsafe fn _mm_cvtsi64_si128(a: i64) -> __m128i
```

Available on **x86-64** and target feature **sse2** only.

[–] Returns a vector whose lowest element is `a` and all higher elements are `0`.

[Intel's documentation](#)

貌似是intel的内在函数，然后图中的这个函数说明，和本应该用的功能对不上，即不能实现从64位整型到双精度浮点数的转换，因此我认为是这里的函数使用错了。

修改解释

于是我又搜了一下其他类似的函数，比如

_mm_cvtsi64x_ss

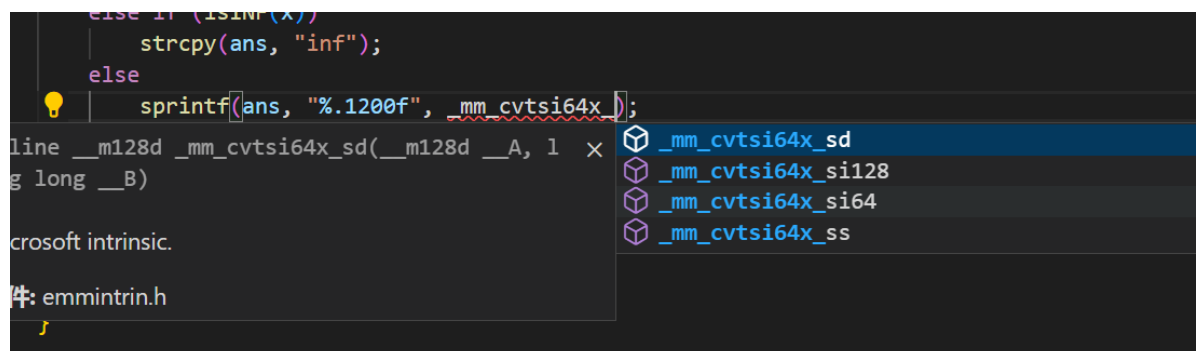
项目 • 2022/09/27 • 6 个参与者

 反馈

Microsoft 专用

生成将 64 位整数转换为标量单精度浮点值的 x64 扩展版本 (`cvtsi2ss`) 指令。

于是我猜测si64x代表64位整数，softDouble代码这里应该是`mm_cvtsi64x_xx`，即左边是si64x，右边是表示双精度浮点数的符号。因为网上相关的资料比较散杂，所以我利用了一下vscode的“联想功能”（如下图），显示了几个选项，依次试了之后发现 `_mm_cvtsi64x_si64` 是所需的正确函数。



第三处错误

找出bug的过程和原因分析

在将上面两个bug找到后，程序基本能正常输出了，但是还是有很多测试点通不过，因此我考虑先从一些极端情况入手。当输入 $-1/0$ 时，程序输出了 `inf`（本应该是 `-inf`），于是我又多试了几个数据，发现程序只能输出 `inf` 而不能输出 `-inf`，于是我先找到 `write_to_string` 函数的定义部分，发现当 `x` 为无穷时只会将 `inf` 拷贝给 `ans` 而没有 `-inf`，因此这里需要补上对 `x` 符号的判断，符号位为 0 就是 `inf`，符号位为 1 就是 `-inf`。

改了之后发现还是不对，因为是输出的除法，于是我又去看divide函数，发现函数在判断除数为0的地方，只返回了INF，因此这里也需要进行补充inf的情况。

修改解释

- 1.在 `write_to_string` 函数中，当x为无穷时，添加对x的符号位的判断；
- 2.在divide函数中，设一个uint64_t变量，`uint64_t sign2 = (lhs ^ rhs) >> 63;`，表示lhs和rhs的符号是否相同，然后在rhs为0、lhs不为0的分支语句中加上对符号位的判断，如果sign2==0，返回INF，否则返回NINF。

第四处错误

后面两处错误比较难找，试了很多数据都检查不出来，包括一些极端的测试点，于是只能从函数逻辑入手，分析是否有逻辑漏洞。

在检查add函数时，先梳理了一下函数的思路，前面几个if语句在判断NaN、无穷的情况，后面基本上是浮点数相加的过程，阶码对齐、有效位相加、将和规格化、舍入、检查溢出，在Rounding中，因为之前先左移了一位，离原位差一位，如果ansf的最低位是0，则直接右移一位，不需要舍入；如果最低位为1，先右移一位回到原位，然后判断是否需要舍入。对比substract函数的Rounding的逻辑，因为之前先左移了2位，所以在Rounding时要右移2位复位，这里和add函数是对应的，substract函数这里是将 $(ansf \& 3)$ 与 2 进行比较，小于2的话就直接右移2位，大于2就直接舍入，等于2就向偶数舍入。对比可知add函数Rounding部分向偶数舍入的情况讨论不全面，要在里面的if中加上去。

第五处错误

是一个比较容易忽视的bug，即移位的位数是否在最大范围内，之前写datalab的时候也出现过类似的错误。之前助教师兄在课程群发过一张图片，“不要假定移更多位可以把位给全都移出去，也不要假定左移负数就变成右移”，对我有所启发，于是着重关注这方面的错误，发现，

The value of `a << b` is the unique value congruent to $a * 2^b$ modulo 2^N where N is the number of bits in the return type (that is, bitwise left shift is performed and the bits that get shifted out of the destination type are discarded). (since C++20)

The value of `a >> b` is $a/2^b$, rounded down (in other words, right shift on signed `a` is arithmetic right shift).

In any case, if the value of the right operand is negative or is greater or equal to the number of bits in the promoted left operand, the behavior is undefined.

In [overload resolution against user-defined operators](#), for every pair of promoted integral types L and R, the following function signatures participate in overload resolution:

加法和减法函数中，计算出ediff后，在之后的while循环中，直接写了 `LowBit(rhsf) >> ediff`，而没有先判断ediff是否大于等于64（等于的话就不移位），因此当ediff恰好等于64时，LowBit(rhsf)中的1本应该被移出去，却没有被移走，cur本应该等于0，在这里却不为0，从而出现逻辑错误。

下面是调试过程，输入 $1 + 2^{64}$ ，ediff等于64，191行执行后，cur等于 `LowBit(rhsf)`，而不为0

没有配置

softDouble.cpp 1 x softDouble.cpp

memory.bin x

0x0020000000000000 > memory.bin

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
000000A0
000000B0
000000C0

Locals

cur: 0
ediff: 64
ans: 0
roundup: false
ansexp: 1087
rhsf: 90071992547...
ansf: 90071992547...
lhs: 489541279495...
rhs: 460718241880...

Registers

调用堆栈

断点

■ All C++ Exceptio...

8num_bug.cpp 90
8num.cpp 110
antbuster.cpp 111
softDouble... 181
softDouble... 384
xorsum.cpp 19
xorsum.cpp 43

182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

uint64_t ans = 0;
bool roundup = false;
uint64_t ansexp = Exp(lhs);
uint64_t rhsf = Fraction(rhs) << 1;
uint64_t ansf = Fraction(lhs) << 1;

while (rhsf != 0)
{
 uint64_t cur = LowBit(rhsf) >> ediff;
 /*uint64_t cur = LowBit(rhsf) >> (ediff
 cur = cur >> (ediff >= 64 ? 1 : 0);*/
 if (cur == 0)
 {
 roundup = true;
 }
 else
 {
 ansf += cur;
 rhsf -= LowBit(rhsf);
 }
}

问题 1 输出 调试控制台 终端

C:\vscode_c\homework\ICS_1> cmd /C "c:\Users\Rachel\.vscode\extensions\ms-vscode.cpptools-1.12.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe --stdin=Microsoft-MIEngine-In-0iscznwe.mes --stdout=Microsoft-MIEngine-Out-tv2rnszw.kzn --stderr=Microsoft-MIEngine-Error-0gfaj3jd.rrq --pid=Microsoft-MIEngine-Pid-bdexqar2.nwj --dbgExe=C:\mingw64\bin\gdb.exe --interpreter=mi "

1 + 18446744073709551616