

实验报告

实验名称	实验四：整数安全函数的设计
1. 攻击面分析：	
1) 函数 my_atoi8：	
➤ 输入字符串可能为空、或包含字母、或不符合常规写法，如“00123”；	
➤ 数字不在[-128,127]内。	
➤ 字符串中有既不是英文也不是数字的其他字符。	
2) 有符号整数运算函数：	
➤ 可能出现溢出错误。	
➤ 除法还可能存在除零错误。	
3) 无符号整数运算函数：	
➤ 除了除法以外的运算可能出现回绕错误。	
➤ 除法可能存在除零错误。	
2. 设计思路：	
1) my_atoi8	
流程：	
顺序	步骤
①	若 str 为空，(*of) = -1 返回 0；
②	while 循环去掉空格；
③	判断遇到的第一个字符是否为 ‘-’，如果是，（可能是负数）：继续去掉空格，判断之后第一个字符是否为 ‘0’，若是返回错误；若不是，while 循环读取字符，如果不是数字返回错误，如果是数字且符合要求，则循环相加，判断是否大于 128，若大于返回错误，否则，返回计算结果取负数。
④	判断遇到的第一个字符是否是数字，如果是，则判断第一个数字是否为 0，如果是 0，则判断其后是否还有数字，如果还有数字表示不符合规定，如果没有，表示结果是 0；如果第一个数字不是 0，判断后续是否为其他字符，若符合要求，则循环相加，判断是否大于 127，若大于返回错误，否则，返回计算结果。
⑤	如果遇到的第一个字符不是数字也不是 ‘-’，则返回错误。
2) iAdd_Safe	
判断 a > INT8_MAX - b a < INT8_MIN - b，如果是则溢出，*of = -2，返回 0，否则返回 a+b。	
3) iSubtract_Safe	
判断 a > INT8_MAX + b a < INT8_MIN + b，如果是则溢出，*of = -2，返回 0，否则返回 a-b。	
4) iMultiply_Safe	
顺序	步骤

①	a>0 且 b>0, 如果 $a > \text{INT8_MAX} / b$, 则溢出错误, *of = -2, 返回 0;
②	a>0 且 b<0, 如果 $(b < \text{INT8_MIN} / a)$, 则溢出错误, *of = -2, 返回 0;
③	a<0 且 b>0, 如果 $a < \text{INT8_MIN} / b$, 则溢出错误, *of = -2, 返回 0;
④	如果 $(a \neq 0)$ 并且 $(b < (\text{INT8_MAX} / a))$ 则溢出错误, *of = -2, 返回 0;
⑤	若以上情况都不符合, 则 *of = 0, 返回 a*b。

5) iDivide_Safe

b 若为 0 则除零错误, *of = -1, 返回 0; 否则判断分子是否为最小值 INT8_MIN 且 b == -1, 如果是, 则溢出, *of = -2, 返回 0, 否则 *of = 0, 返回 a/b。

6) Add_Safe

判断 $\text{UINT8_MAX} - a < b$ 是否成立, 成立则表示会回绕, *of = -2, 返回 0; 否则 *of = 0, 返回 a+b。

7) Subtract_Safe

判断 $a < b$ 是否成立, 成立则表示会回绕, *of = -2, 返回 0; 否则 *of = 0, 返回 a-b。

8) Multiply_Safe

判断 $a > \text{UINT8_MAX} / b$ 是否成立, 成立则表示会回绕, *of = -2, 返回 0; 否则 *of = 0, 返回 a*b。

9) Divide_Safe

无符号整数除法不产生回绕, 仅在 $b == 0$ 时产生除零错误, *of = -1, 返回 0; 否则 *of = 0, 返回 a/b。

10) 结果 (按照实验所给用例输出):

➤ my_atoi8:

```

第一组:
(*of) == 0
(*of) == 0
(*of) == 0
(*of) == 0
第二组:
(*of) == -2
(*of) == -2
(*of) == -2
第三组:
(*of) == -1
(*of) == -1
(*of) == -1
(*of) == -1
第四组:
(*of) == -1
(*of) == -1
(*of) == -1
第五组:
(*of) == -2
(*of) == -2
第六组:
(*of) == 0
(*of) == 0
(*of) == 0

```

➤ 有符号运算:

有符号整数加减乘除 加法第一组: 12+13 = 25 (*of)==0 127+0 = 127 (*of)==0 -12+-13 = -25 (*of)==0 -127+13 = -114 (*of)==0 第二组: 127+3 = 溢出错误! (*of)==-2 -126+-13 = 溢出错误! (*of)==-2	减法第一组: 12-13 = -1 (*of)==0 127-0 = 127 (*of)==0 -12--13 = 1 (*of)==0 -127-1 = -128 (*of)==0 第二组: 127--1 = 溢出错误! (*of)==-2 -126-13 = 溢出错误! (*of)==-2
乘法第一组: 2*30 = 60 (*of)==0 127*1 = 127 (*of)==0 -2*30 = -60 (*of)==0 -12*-3 = 36 (*of)==0 -128*1 = -128 (*of)==0 第二组: 127*3 = 溢出错误! (*of)==-2 -28*6 = 溢出错误! (*of)==-2 -128*-1 = 溢出错误! (*of)==-2	除法第一组: 127/3 = 42 (*of)==0 -28/7 = -4 (*of)==0 第二组: -128/-1 = 溢出错误! (*of)==-2 第三组: 127/0 = 除零错误! (*of)==-1 -28/0 = 除零错误! (*of)==-1 -128/0 = 除零错误! (*of)==-1

➤ 无符号整数运算:

无符号整数加减乘除 无符号加法第一组: 12+13 = 25 (*of)==0 255+0 = 255 (*of)==0 132+13 = 145 (*of)==0 第二组: 255+1 = 回绕错误! (*of)==-2 124+123 = 247 (*of)==0 无符号减法第一组: 17-13 = 4 (*of)==0 127-0 = 127 (*of)==0 232-13 = 219 (*of)==0 第二组: 127-128 = 回绕错误! (*of)==-2 2-255 = 回绕错误! (*of)==-2	无符号乘法第一组: 2*30 = 60 (*of)==0 127*1 = 127 (*of)==0 122*2 = 244 (*of)==0 第二组: 127*3 = 回绕错误! (*of)==-2 28*26 = 回绕错误! (*of)==-2 无符号除法第一组: 234/3 = 78 (*of)==0 228/7 = 32 (*of)==0 第二组: 127/0 = 除零错误! (*of)==-1 0/0 = 除零错误! (*of)==-1
--	--

3. 心得体会

做完本次实验, 了解了 int8_t、int16_t、int32_t 等数据类型, 以及溢出、回绕和除零错误。知道如何去判断有符号数溢出和无符号数回绕的问题以及除零问题, 尤其是有符号数的乘除。设计计算机关于各种整数的计算规则应当十分谨慎。同时也明白以后编程应当小心使用各种类型, 防止出现错误。