软件安全实验

参考要求

2018 年 4 月 25 日

目录

实验提交要求：

1、提交整个工程文件，SafeCodeProject；

2、工程文件压缩打包后，用自己学号+姓名命名，如“20170906435_汪海洋”；

3、压缩包统一交本班班长，发送到邮箱：zhanghuixiang@nwpu.edu.cn

实验一：字符串函数的使用

一、实验目标

1. 第二章-字符串

2. 掌握常用的字符串函数；

3. 掌握典型的 IDE 环境（VS2017 社区版）的使用。

二、内容与要求

1. 所在模块：StrSafe

2. 编写函数：**int isUsernameOK(char* name);**

功能：判断 name 字符串是否符合要求：name 只允许数字或字母，不小于 8 个字符，不超过 16 个字符，且要求以字母开头。

参数 name：C 字符串。

返回值：符合要求返回 0，否则返回-1.

提示：可使用 isalnum，isalpha 等判断。

3. 编写函数：**int isPasswordOK(char* pwd);**

功能：判断 pwd 字符串是否符合要求：允许数字或字母，不小于 8 个字符，不超过 16 个字符，至少包含一个数字和一个字母，且至少一个大写字母。

参数 pwd：C 字符串。

返回值：符合要求返回 0，否则返回-1.

提示：使用 isdigit，isupper，isalpha 等判断。

4. 在 main 函数中对所编写函数进行测试，说明所使用的测试用例。

5. MSDN: https://msdn.microsoft.com/en-us/library/hh875057.aspx

6. 建议学时：1 学时。

三、参考测试向量

序号	**int isUsernameOK(char* name);**
1	**isUsernameOK(“a12345678”)**
2	**isUsernameOK(“a123”)**
3	**isUsernameOK(“a123bcdefg4567890”)**

4	**isUsernameOK(“”)**
5	**isUsernameOK(0)**
6	**isUsernameOK(“123456789”)**
7	**isUsernameOK(“abcdefghjik”)**

序号	**int isPasswordOK(char* pwd);**
1	**isPasswordOK(“1234A5678”)**
2	**isPasswordOK(“a123”)**
3	**isPasswordOK(“a123Bcdefg4567890”)**
4	**isPasswordOK(“”)**
5	**isPasswordOK(0)**
6	**isPasswordOK(“123456789”)**
7	**isPasswordOK(“abcdefghjik”)**
8	**isPasswordOK(“1234abcde8”)**

实验二：字符串函数的设计

一、实验目标

1. 第二章-字符串

2. 掌握常用的字符串函数的设计；

3. 理解字符串处理函数中存在的安全缺陷。

二、内容与要求

1. 所在模块：StrSafe

2. 编写函数：**int gets_safe(char *str, rsize_t n);**

功能：从 stdin 读入用户输入，直到用户敲入 Enter 终止。

（1）如果用户输入的字符个数大于等于 n，则对输入字符串进行截断后保存前(n-1)个字符，最后一个位置置为空字符，以保证 str 指向的字符串符合 C 语言定义。同时，要求对 stdin 中缓冲的多余字符进行清空。

（2）如果输入的字符个数小于 n，则保存所有字符，最后附加空字符，以保证 str 指向的字符串符合 C 语言定义。

（3）本函数不保存回车换行符。

（4）如果读取中出现错误，则 str[0] 置零，返回-2。

（5）如果参数错误（如空指针），则 str[0] 置零（如果确定 str 非空），返回-1.

参数 str：指向本地缓冲区的指针，用户输入的数据将被保存在该缓存区内。

参数 n：本地缓冲区的大小。

返回值：读入的字符个数，不包括空终止符。出现读取错误，返回-2，参数错误返回-1.

提示：使用 getchar()获取用户输入的单个字符，ferror 函数测试读取错误。

3. 编写函数：**int strcpy_safe(char * dest, rsize_t destsz, const char * src);**

功能：复制源字符串 src 到目的缓冲区 dest。

（1）如果 src 字符串长度大于等于 destsz，则只复制前 (destsz-1) 个字符，目的缓冲区最后一个字符置零，保证 dest 字符串符合 C 语言对字符串的要求。

（2）如果 src 字符串长度小于 destsz，则复制整个源字符串，最后附加空字符，以保证 dest 字符串符合 C 语言对字符串的要求。

（3）如果参数错误，则 dest[0] 置零（如果确定 dest 非空），返回-1.

参数 dest：指向目的缓冲区的指针。

参数 destsz：目的缓冲区的大小。

参数 src：指向源字符串的指针。

返回值：实际复制的字符个数，不包括空终止符。出现错误返回-1.

提示：使用指针实现单个字符逐一复制。

4. 在 main 函数中对所编写函数进行测试，说明所使用的测试用例。

5. MSDN: https://msdn.microsoft.com/en-us/library/hh875057.aspx

6. 建议学时：2 学时。

三、参考测试向量

序号	**int gets_safe(char *str, rsize_t n);**
1	**Char buf[8];** **gets_safe(buf, sizeof(buf));** **Stdin: “123456”, “1234567”, “12345678”, “\n”**
2	**Char* buf = 0;** **gets_safe(buf, sizeof(buf));**
3	**Char* buf[8];** **gets_safe(buf, 0);**

序号	**int strcpy_safe(char * dest, rsize_t destsz, const char * src);**
1	**Char dest[8];** **Char *src = “12345”;** **strcpy_safe(dest, sizeof(dest), src);**
2	**Char dest[8];** **Char *src = “1234567”;** **strcpy_safe(dest, sizeof(dest), src);**
3	**Char dest[8];** **Char *src = “12345678”;**

	strcpy_safe(dest, sizeof(dest), src);
4	**Char *src = “1234567”;** **strcpy_safe(0, 8, src);**
5	**Char dest[8];** **strcpy_safe(dest, sizeof(dest), 0);**
6	**Char dest[8];** **Char *src = “12345678”;** **strcpy_safe(dest, 0, src);**

实验三：C 内存管理程序设计

一、实验目标

1. 第四章-C 内存管理；

2. 了解动态内存管理的基本原理；

3. 掌握动态内存分配函数的使用方法。

二、内容与要求

1. 所在模块：**MemPool**

2. 编写一个简易的内存池管理模块，采用 C 内存管理函数实现内存池的管理。内存池按块进行管理，每块内存为 256 字节。MemPool 完成内存池的初始化、内存分配、内存回收。内部实现可自由发挥，但函数接口必须一致。

3. 编写函数：**int initPool(size_t size = 1000);**

功能：分配初始内存池，内存池初始大小默认为 1000 个块，即 256 x 1000 字节。

参数 size：内存池初始化时分配的内存块个数，默认为 1000 块。

返回值：成功完成 1000 块内存的申请则返回 0，否则返回-1.

提示：可使用 malloc、calloc 等函数完成内存初始化。

4. 编写函数：**char* allocBlock();**

功能：从内存池中找到一块空闲内存块进行分配。分配的内存被初始化为 0。**如果内存池内存已分配完，需对现有内存池进行扩充。**

返回值：返回分配的内存块首地址，如果出现错误则返回空指针。

提示：可使用 malloc、calloc 函数、memset 函数。注意如果使用 realloc 扩充内存，通过 **allocBlock** 已经分配出去的内存块地址可能会无效。

5. 编写函数：**int freeBlock(char* buf);**

功能：将 buf 指向的内存块释放，该内存块并不执行堆内存的释放操作，只是放回内存池供下次分配使用。释放时，需对 buf 指向的内存区域**进行清零操作**。

参数 buf：指向要放回内存池中的内存块。

返回值：执行成功返回 0，buf 指针参数不合规返回-1，未找到对应的块返回-2.

提示：可使用 memset 函数，或者 Windows API SecureZeroMemory 函数清零。

6. 编写函数：**int freePool();**

功能：将整个内存池的内存全部释放，首先判断是否存在未回收的内存块，如果有则需返回-1，如果没有则对内存池所有内存块进行清零，然后执行 free 操作。.

返回值：执行成功返回 0，如果存在未回收的内存块，则需返回-1.

提示：可使用 free、memset 函数。

7. 编写函数：**int freePoolForce();**

功能：无论是否存在未回收的内存块，都将整个内存池的内存块进行清零，然后执行 free 操作。.

返回值：执行成功返回 0，出现错误，则返回-1.

提示：可使用 free、memset 函数。

8. 编写函数：**int getBlockCount();**

功能：返回内存池中所有的内存块个数。

9. 编写函数：**int getAvailableBlockCount();**

功能：返回内存池中空闲可分配的内存块个数。

10. 在 main 函数中对所编写函数进行测试，说明所使用的测试用例。

11. MSDN: https://msdn.microsoft.com/en-us/library/hh875057.aspx

12. 建议学时：2 学时。

三、参考测试向量

序号	
1	**initPool(8);allocBlock();char* buf = allocBlock();** **buf[0] + buf[1] + ... +buf[255] = 0**
2	**getBlockCount()==8;getAvailableBlockCount()==6;** **allocBlock();allocBlock();getAvailableBlockCount()==4;**
3	**freeBlock();getBlockCount()==8;getAvailableBlockCount()==5;**
4	**freeBlock(buf); buf[0] + buf[1] + ... +buf[255] = 0**
5	**freePool()==-1;freePoolForce();** **getBlockCount()==0;getAvailableBlockCount()==0;**

实验四：整数安全函数的设计

一、实验目标

1. 第二章-字符串、第五章-整数安全

2. 掌握常用的字符串函数；

3. 理解整数溢出问题。

二、内容与要求

1. 所在模块：MathSafe

2. 编写函数：**int8_t my_atoi8(char* str, int* of);**

功能：将 str 指向的整数字符串转换为一个整数类型。本函数只针对[-128,127]范围内的整数执行转换。

参数 str：C 字符串。

参数 of：整数指针，返回转换中的错误情况。Str 合规且转换成功，(*of) = 0；str 不指向数字或者为空，(*of) = -1；str 指向的数字不合法，(*of) = -2.

返回值：返回转换后的整数，如果参数 str 不正确或者指向的数字越界，则返回 0.

提示：p164 例程改造。

3. 编写函数：

（1）int iAdd_Safe(int8_t a, int8_tb, int* of);　功能：a+b

（2）int iSubtract_Safe(int8_t a, int8_tb, int* of); 功能：a-b

（3）int iMultiply_Safe(int8_t a, int8_tb, int* of); 功能：a*b

（4）int iDivide_Safe(int8_t a, int8_tb, int* of); 功能：a/b

功能：分别实现有符号数 int8_t 的加减乘除运算。

参数 of：整数指针，返回计算中的错误情况。(*of) = 0 计算成功；(*of) = -1 表示除零错；(*of) = -2 表示溢出错。

返回值：计算成功返回计算结果，否则返回 0.

4. 编写函数：

（1）UINT Add_Safe(uint8_t a, uint8_tb, int* of);　功能：a+b

（2）UINT Subtract_Safe(uint8_t a, uint8_tb, int* of); 功能：a-b

（3）UINT Multiply_Safe(uint8_t a, uint8_tb, int* of); 功能：a*b

（4）UINT Divide_Safe(uint8_t a, uint8_tb, int* of); 功能：a/b

功能：分别实现无符号数 uint8_t 的加减乘除运算。

参数 of：整数指针，返回计算中的错误情况。(*of) = 0 计算成功；(*of) = -1 表示除零错；(*of) = -2 表示回绕错。

返回值：计算成功返回计算结果，否则返回 0.

5. 在 main 函数中对所编写函数进行测试，说明所使用的测试用例。

6. MSDN: https://msdn.microsoft.com/en-us/library/hh875057.aspx

7. MSDN: SafeInt Library, https://msdn.microsoft.com/en-us/library/dd570023.aspx

8. 建议学时：2 学时。

三、参考测试向量

序号	**int8_t my_atoi8(char* str, int* of);**	
1	**my_atoi8(“-128”), my_atoi8(“127”), my_atoi8(“0”), my_atoi8(“-1”)**	(*of) = 0
2	**my_atoi8(“1285”), my_atoi8(“-129”), my_atoi8(“130”)**	(*of) = -2
3	**my_atoi8(“”), my_atoi8(0), my_atoi8(“abc”), my_atoi8(“12a”)**	(*of) = -1
4	**my_atoi8(“123a”), my_atoi8(“12a”), my_atoi8(“-12a”)**	(*of) = -1
5	**my_atoi8(“00123”), my_atoi8(“-00123”)**	(*of) = -2
6	**my_atoi8(“ 123 ”), my_atoi8(“-23 ”), my_atoi8(“-2 3 ”)**	(*of) = 0

序号	**有符号整数加减乘除**	
1	**iAdd_Safe(12,13), iAdd_Safe(127,0),** **iAdd_Safe(-12, -13), iAdd_Safe(-127, 13)**	(*of) = 0
2	**iAdd_Safe(127,3), iAdd_Safe(-126,-13),**	(*of) = -2
3	**iSubtract_Safe(12,13), iSubtract_Safe(127,0),** **iSubtract_Safe(-12, -13), iSubtract_Safe(-127, 1)**	(*of) = 0
4	**iSubtract_Safe(127,-1), iSubtract_Safe(-126,13),**	(*of) = -2
5	**iMultiply_Safe(2,30), iMultiply_Safe(127,1), iMultiply_Safe(-2,30),** **iMultiply_Safe(-12, -3), iMultiply_Safe(-128, 1)**	(*of) = 0

6	**iMultiply_Safe(127,3), iMultiply_Safe(-28,6), iMultiply_Safe(-128,-1),**	(*of) = -2
7	**iDivide_Safe(127,3), iDivide_Safe(-28,7), iDivide_Safe(-128,-1),**	(*of) = 0
8	**iDivide_Safe(-128,-1),**	(*of) = -2
9	**iDivide_Safe(127,0), iDivide_Safe(-28,0), iDivide_Safe(-128,0),**	(*of) = -1

序号	**无符号整数加减乘除**	
1	**Add_Safe(12,13), Add_Safe(255,0), Add_Safe(132, 13)**	(*of) = 0
2	**Add_Safe(255,1), Add_Safe(124,123)**	(*of) = -2
3	**Subtract_Safe(17,13), Subtract_Safe(127,0), Subtract_Safe(232, 13)**	(*of) = 0
4	**Subtract_Safe(127,128), Subtract_Safe(2,255)**	(*of) = -2
5	**Multiply_Safe(2,30), Multiply_Safe(127,1), Multiply_Safe(122,2),**	(*of) = 0
6	**Multiply_Safe(127,3), Multiply_Safe(28,26)**	(*of) = -2
7	**Divide_Safe(234,3), Divide_Safe(228,7)**	(*of) = 0
8		(*of) = -2
9		(*of) = -1

实验五：多线程程序练习

一、实验目标

1. 第六章-并发
2. 掌握多线程程序编写方法；
3. 理解并发中的竞争问题。

二、内容与要求

1. 所在模块：LogSafe

2. 编写函数：**int initLog(char* path);**

功能：初始化日志模块，设置日志文件存放的路径。

参数 path：指向一个文件路径的 C 字符串。

返回值：日志文件创建成功返回 0，否则返回-1.

提示：可使用 C 语言中文件 IO 操作。

3. 编写函数：**int logStr(char* level, char* str);**

功能：将字符串 str 输出到日志文件中，根据 level 字符串设置输出的日志信息。日志格式：Level || 线程 id || 时分秒 || str 字符串。要求实现线程安全，确保多线程使用时输出不产生乱序或截断等。当日志文件大于 1MB 时，需执行备份操作。

参数 level：C 字符串，可能的 level 包括：Normal，Warning，Alert，Error 等。

参数 str：C 字符串，用户需要输出的日志信息。

返回值：符合要求返回 0，否则返回-1.

提示：使用标准线程库，可使用互斥量保证线程安全，尝试 RAII 用法。

4. 在 main 函数中对所编写函数进行测试，说明所使用的测试用例。

5. MSDN: https://msdn.microsoft.com/en-us/library/hh875057.aspx

6. 建议学时：2 学时。

三、参考测试向量

序号	
1	initLog(“C:/temp/log.txt”); CreateThread1, CreateThread2, CreateThread3
2	ThreadFun () {

```
  logStr(“Normal”, “Start Normal Testing ...”);
  logStr(“Warning”, “Start Warning Testing ...”);
  logStr(“Alert”, “Start Alert Testing ...”);
  logStr(“Error”, “Start Error Testing ...”);
}
```

实验六：C 相似性计算

一、实验目标

1. 软件相似性

2. 掌握 N-Gram 算法；

3. 理解基于文本的软件相似性比较原理。

二、内容与要求

1. 所在模块：SoftwareSafe

2. 编写函数：**int removeAnnotation(char* path);**

功能：去除 C 文件中包含的注释信息，即 /* */，//所带的注释文本被清除，但不能影响 C 源代码的格式，即只去注释，不改格式。

参数 path：指向一个 C 源文件路径的 C 字符串。

返回值：执行成功返回 0，如果错误返回-1.

3. 编写函数：**double computeTextSimularity(char* path1, char* path2, int k);**

功能：计算两个文本文件的相似度。

参数 path1：指向一个文本文件路径的 C 字符串。

参数 path2：指向一个文本文件路径的 C 字符串。

参数 k：N-Gram 切片的值。

返回值：执行成功返回两个文件的文本相似度，如果计算错误返回-1.

提示：去除注释、空白字符、控制字符等，形成一个长字符串再切片比较。

4. 编写函数：**double computeBinSimularity(char* path1, char* path2, int k);**

功能：计算两个二进制文件的相似度。

参数 path1：指向一个二进制文件路径的 C 字符串。

参数 path2：指向一个二进制文件路径的 C 字符串。

参数 k：N-Gram 切片的值。

返回值：执行成功返回两个文件的二进制相似度，如果计算错误返回-1.

提示：按二进制读入 01 序列，切片进行比较。

5. 在 main 函数中对所编写函数进行测试，说明所使用的测试用例。

6. MSDN: https://msdn.microsoft.com/en-us/library/hh875057.aspx

7. 建议学时：4 学时。

三、参考测试向量

序号	
1	
2	