



|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | 18070100017 | **姓名** | 钟保明 |
| **班级** | **1803053** | **任课教师** | **张淑平** |
| **实验名称** | 简单类型、表达式和基本控制结构 | | |
| **实验学期** | **2019 – 2020 学年第2学期** | | |
| **实验日期** | 2020年3月12日 | **实验地点** |  |
| **报告成绩** |  | | |

西安电子科技大学计算机科学与技术学院

# 实验目的

1. 计算名字的相关数据,熟悉c++的输入输出，学习关联数组map的使用；
2. 编写一个可将给定的字符串转换为(机内)整数的函数atoi,并且支持十进制整数字面量、八进制字面量、十六进制字面量和c++字符字面量的各种形式。
3. 编写一个将整数转换成字符串的函数itoa,可以按照十进制转换、八进制转换和十六进制转换。
4. 编写一个简单的桌面计算器，实现加减乘除赋值运算。

# 实验环境

操作系统：Windos 10

开发工具：Visual Studio 2019

# 实验内容

## 计算姓名相关的数据

输入数据是一系列 name 和 value 构成的**值对**，其中name是中间没有空白字符的单词（字符串），value是算术数值（整数、实数均可），对于每个name，计算该name在输入中对应的所有value的总和、平均数；计算所有name对应的value的总和、平均数。

## 定义函数 atoi()

编写一个可将给定的字符串转换为(机内)整数的函数，其接口应为：int atoi ( const char\* ); 符串文本的书写形式应支持十进制整数字面量、八进制整数字面量、

十六进制整数字面量、十六进制整数字面量。

## 定义函数 itoa()

编写char\* itoa(int i, char b[]);函数将指定的整数i转换为字符串，并将所得字符 串存储到参数b指定的字符数组中。

## 编译并运行第六章的桌面计算器程序

编写一个桌面计算器，实现加减乘除和赋值操作。

# 数据结构与算法设计

## 计算名字的相关数据

**模块结构及文件组织设计：**

模块1：主控模块，仅包括文件run.cpp ，定义了 main()函数。

模块2：信息模块，包括以下两个文件：

Info.h 定义了每个姓名对应的信息；

Info.cpp 实现类Info的构造函数。

**关键数据结构设计：**

数据结构1：定义了每个姓名对应的信息Info，其成员 sum 存放name对应的value之和，count对应name出现的次数

class Info

{

public:

double sum;// 存放name对应的value

int count;//name出现的次数

Info();//构造函数

};

全局变量：map<string, Info> table; ////通过全局变量table来储存输入的数据。

**算法1.1 int main()**

作 用：主控函数，也实现对题目所需其他内容的测试。

参 数：无参数。

返回值：总是返回0。

计算过程：

1. 循环读入name及其value值，table[name].sum加上这次输入的value值，table[name].count自增;
2. 输出每个name的value的总和和平均值,并计算all\_sum(即所有name的sum之和),all\_count(即所有name出现的次数之和)。
3. 输出所有 name 的所有 value 之和、平均数。

**算法1.2 Info::Info()**

作 用：Info类的构造函数，初始化sum为0，初始化count为0

参 数：无参

返回值：构造函数无返回值

计算过程：

sum = 0;

count = 0;

## 本题的名称

**模块结构及文件组织设计：**

模块1：主控模块，仅包括文件run.cpp ，定义了 main()函数。

模块2：信息模块，包括以下两个文件：

atoi.h 定义了命名空间，枚举类型Classify,以及相关函数接口。

atoi.cpp 实现了ato.h的函数接口。

**关键数据结构设计：**

数据结构设计1：

enum Classify

{

Oct = 8, //八进制整数字面量

Dec = 10,//十进制整数字面量

Hex = 16,// 十六进制整数字面量

Error = 0 //非法的字符串

};

**算法2.1 int main()**

作 用： 主控函数，也实现对题目所需其他内容的测试

参 数： 无参

返回值： 返回0

计算过程：

1.输入需要转换的字符串，储存在ch中，

2.调用my\_atoi::atoi(ch)函数，得到结果并输出。

**算法2.2 int my\_atoi::atoi(const char \*str)**

作 用： 根据传入的字符串，返回相应的整数

参 数： 需要转换的字符串常量str;

返回值： str转换得到的整数。

计算过程：

1. 调用transform函数进行预处理，

2. 调用classify函数进行分类，

3. 根据分类结果选择调用decinal、octonary、hex、handle\_error函数，

4. 判断得到的结果是否符合要求，如果不符合则改正。

5. 返回结果。

**函数2.3 string my\_atoi::tranform(****const char \*str, int \*symbol)**

作用：

1. 去除空白符，

2. 判断正负号，

3. 将大写字符转换为小写字符，

4. 将c++的各种字符字面量转换成相应的进制字符。

参数：

const char \*str, 需转换的字符常量

int \*symbol 存放正负信息

返回值：

返回转换后的字符，string 类型

计算过程：

1. 定义变量string string1；指示下标的变量i,

2. 将i定位到第一个非空白符位置处，

3. 如果有负号，则\*symbol=-1,如果没有正负号或者有正好，则\*symbol = 1.

4. 如果不是\x \0这种字符，直接将它push\_back到string1中(大写转小写)，反之则将\x \0后面的两个字符转换成十进制字符，并放到string1中。

5. 返回string1。

**函数2.4 my\_atoi::Classify my\_atoi::classify(const char \*str)**

作用：

1. 判断str字符串属于哪一种进制字符，或者是否有错误。

参数：

const char \*str, 需转换的字符常量

返回值：

返回枚举变量中的一种，表示属于哪个种类型。

计算过程：

1. 如果第一个字符不为0，并且所有字符都是数字字符，则返回Dec，

2. 如果第一个字符为0，第二个字符不为x，其他的字符都是0-7，返回Oct，

3. 如果第一个字符为0，第二个字符为x,其他的字符为十六进制字符，返回Hex,

4. 最后返回Error(表示其他情况)。

**函数2.5 long long my\_atoi::decimal(const char \*str)**

作用：

将str转换成十进制长整型数字，

参数：

const char \*str, 需转换的字符常量，

返回值：

转换后的长整型数字。

计算过程：

从左到右地扫描字符串，边扫描边计算/转换。

**函数2.6 long long my\_atoi::octonary(const char \*str**

作用：

将str转换成八进制长整型数字，

参数：

const char \*str, 需转换的字符常量，

返回值：

转换后的长整型数字。

计算过程：

从左到右地扫描字符串，边扫描边计算/转换。

**函数2.7** **long long my\_atoi::hex(const char \*str)**作用：

将str转换成十六进制长整型数字，

参数：

const char \*str, 需转换的字符常量，

返回值：

转换后的长整型数字。

计算过程：

从左到右地扫描字符串，边扫描边计算/转换。

**函数2.8 long long my\_atoi::handle\_error(const char \*str)**

作用：

处理错误

参数：

const char \*str, 需转换的字符常量

返回值：

转换后的长整型数字。

计算过程：

1. 字符串长度为0，返回0，

2. 第一个字符非数字，返回0，

3. 没有一个数字字符返回0，

4. 截取第一个非数字符前面的字符，返回其十进制转换结果，

5. 如果有字符e，则将其前面的数字字符串调用decimal转换成十进制数a，调用fetcher\_power取得指数，调用decimal函数转换成十进制数b，如果a \*10^ b超过了int范围或者为0，则返回a，反之返回a \*10^ b。

6. 返回0。

**函数2.9 long long my\_atoi::fetch\_power(string string1, int pos)**

作用：

返回e后面的指数

参数：

字符串string1,字符e在string1的下标

返回值：

转换后的长整型数字。

计算过程：

将pos后面的字符串取下，调用decimal函数返回结果。

## 定义函数 atoi()

**模块结构及文件组织设计：**

模块1：主控模块，仅包括文件run.cpp ，定义了 main()函数。

模块2：信息模块，包括以下两个文件：

itoa.h 定义了命名空间，声明了函数接口。

itoa.cpp 实现了itoa.h的函数接口。

**算法3.1 int main()**

作 用： 主控函数，也实现对题目所需其他内容的测试

参 数： 无参

返回值： 返回0

计算过程：

1.输入需要转换的整数，和进制基数，

2.调用itoa函数，得到结果并输出。

**算法3.2 char \*my\_itoa::itoa(int i, char b[], int base = 10)**

作 用： 根据传入的整数，返回相应的字符串首地址,

参 数： 需要转换的整数i，基数base,

返回值： 相应的字符串首地址。

计算过程：

1. 如果base = 10(默认) 调用decimal函数得到结果，

2. 如果base = 16 调用hex函数得到结果，

3. 如果base = 8 调用octonary函数得到结果，

4. 返回结果。

**函数3.3** **long long my\_itoa::complent(int num)**

作用：

负数的补码运算

参数：

原码num

返回值：

返回补码

计算过程：

1. 计算param = 2^32.

2. 返回param+num。

**函数3.4** **int my\_itoa::dec\_to\_hex(long long num, char b[])**

作用：

将十进制整数num转换成十六进制字符串b。

参数：

十进制整数，待储存的十六进制字符串b。

返回值：

十六进制字符串的长度。

计算过程：

1. b[0]=’0’ ,b[1]=’x’,i=2,

2. 取num%16，转换成十六进制字符，放到栈中。

3. num = num/16。++i,

4. 重复(2)(3)直到num=0。

5. 将栈中的元素依次取出放入b中

**函数3.5** **int my\_itoa::dec\_to\_oct(long long num, char b[])**

作用：

将十进制整数num转换成八进制字符串b。

参数：

十进制整数，待储存的八进制字符串b。

返回值：

八进制字符串的长度。

计算过程：

1. b[0]=’0’ ,i=1,

2. 取num%8，转换成八进制字符，放到栈中。

3. num = num/8。++i,

4. 重复(2)(3)直到num=0。

5. 将栈中的元素依次取出放入b中。

**函数3.6**  **int my\_itoa::decimal(int num1, char b[])**

作用：

将十进制整数num转换成十进制字符串b。

参数：

十进制整数，待储存的十进制字符串b。

返回值：

十进制字符串的长度。

计算过程：

1. 将num转换为long long型，如果num<0,令num=-num,b[0]=’-’，

2. 取num%10，放到栈中,

3. num = num/10。++i,

4. 重复(2)(3)直到num=0。

5. 将栈中的元素依次取出放入b中

**函数3.7 int my\_itoa::hex(int num, char b[])**

作用：

将十进制整数num转换成十六进制字符串b。

参数：

十进制整数，待储存的十六进制字符串b。

返回值：

十六进制字符串的长度。

计算过程：

1. 如果num<0，则调用complent函数取num的补码。

2. 调用dec\_to\_hex函数取得num的十六进制字符串。

**函数3.8 int my\_itoa::octonary(int num, char b[])**

作用：

将十进制整数num转换成八进制字符串b。

参数：

十进制整数，待储存的八进制字符串b。

返回值：

八进制字符串的长度。

计算过程：

1. 如果num<0，则调用complent函数取num的补码。

2. 调用dec\_to\_oct函数取得num的八进制字符串。

## 4.4 桌面计算器

**模块结构及文件组织设计：**

模块1：主控模块，仅包括文件run.cpp ，定义了 main()函数。

**关键数据结构设计：**

数据结构设计1：

enum Token {

NAME,NUMBER,END,ERROR,PLUS = '+',MINUS = '-',

MUL = '\*',DIV = '/',PRINT = ';',ASSIGN = '=',LP = '(',RP = ')'

};

全局变量：

const double flag = 6.666666;//出现了这个数字表示这个位置是操作符的位置

stack<char> temp\_stack;//将中缀表达式转换为后缀表达式用的临时栈

stack<double> calcu;//计算后缀表达式时储存数字的栈

vector<double> number;//储存后缀表达式用的向量

vector<char> operate;//储存后缀表达式中的操作符用的向量

map<char, int> m;//储存操作符及其优先级的对应关系

map<string, double> table;//储存变量及其值

Token curr\_tok = PRINT;//储存当前的转态

double number\_value = 0;//储存当前读取的值

string string\_value;//储存当前读取的变量名

**算法4.1 int main()**

作 用： 主控函数，也实现对题目所需其他内容的测试

参 数： 无参

返回值： 返回0

计算过程：

1.初始化pi e的值，初始化操作符的优先级 ，

2.读入一个合法字符，

3.调用dispatch函数取处理当前的这行表达式。

4.如果curr\_token不为ERROR,则输出表达式的结果，否则输出“input error”，

5.将所有栈和向量设置为空。

**算法4.2 Token input()**

作 用： 读取一个合法字符(串)或数字，

参 数： 无，

返回值： 读取的内容的类型，

计算过程：

1. 舍去一开始的空白符，

2. 读取第一个字符，如果是数字字符，退回读取这个数字，如果是字符,退回读取字符串，如果是其他类型则返回其他类型的枚举类型。

**函数4.3** double dispatch()

作用：

根据curr\_toke的类型，进行相应的处理。

参数：

无

返回值：

表达式结果

计算过程：

1. 如果curr\_token为NAME并写现在读取的值为赋值符，则进行赋值。

2. 如果开头出现赋值符，乘号，除号，右括号则令curr\_token=ERROR，返回0.

3. 其他情况则调用express\_value函数求值。

**函数4.4** **double express\_value()**

作用：

求表达式的值。

参数：

无

返回值：

表达式的值，

计算过程：

1. 调用suffix函数求后缀表达式，

2. 调用caculate函数求后缀表达式的值。

**函数4.5** **void suffix()**

作用：

将输入的中缀表达式转换成后缀表达式。

参数：

无

返回值：

无

计算过程：

1. 一直输入，直到curr\_token为END或PRINT。

2. 对输入的每一个操作符，调用to\_stack函数，并传入相应的优先级。

3. 对于输入的每一个数字，直接放到number中。

**函数4.6**  **void to\_stack(char ch, int prior1)**

作用：

根据优先级将读取到的字符放临时栈中，并将临时栈中的字符根据优先级关系放入operate中

参数：

操作符ch，优先级prior1

返回值：

无

计算过程：

1. 如果ch=’)’直接将临时栈中’(’前面的字符全部放入operate中

2. 如果临时栈的栈顶元素的优先级比ch的低，则直接将ch放入临时栈

反之逐个将临时栈的栈顶元素放入operate中，直到栈顶元素的优先级比ch的低或者栈顶为’(或者栈空。

3. 每次将操作符放入operate中时，将flag添加到number中，表示这个位置为操作符。

**函数4.7 double caculate()**

作用：

计算后缀表达式的值

参数：

无

返回值：

表达式的值

计算过程：

1. 临时栈中的操作符全部放入operate中，

2. 从前往后读入数字，如果不是flag则直接放入calcu栈中，如果是flag，则取下calcu栈的两个值，取出operae中的操作符计算，将结果放入calcu中。

3. 返回calcu.top().

# 测试用例与测试结果

## 计算姓名相关的数据

|  |  |  |
| --- | --- | --- |
| 序号 | 输入 | 输出 |
| 1 | Xi 60 wei 50.2  Jin 100 wei 40.5  wei 1.1 Jin 200.0 | 按名字小计：  Xi 总和: 60 平均值：60  wei 总和: 50 平均值：50  总计：  总和：110 平均值：55 |
| 2 | zhang 11 li 22  wang 12 zhong 85  li 58 li 45  wang 69 zhang 415  zhang 12 li 454  ji 12 li 454 | 按名字小计：  ji 总和: 12 平均值：12  li 总和: 1033 平均值：206.6  wang 总和: 81 平均值：40.5  zhang 总和: 438 平均值：146  zhong 总和: 85 平均值：85  总计：  总和：1649 平均值：137.417 |

## 定义函数 atoi()

|  |  |  |
| --- | --- | --- |
| 序号 | 输入 | 输出 |
| 1 | 123 | 123 |
| 2 | -123 | -123 |
| 3 | 0123 | 83 |
| 4 | -0123 | -83 |
| 5 | 0x123 | 291 |
| 6 | +0x123 | 291 |
| 8 | 01\\062\\063 | 83 |
| 9 | 1\\062\\x33 | 123 |
| 10 | 0xF\\x66A | 3946 |
| 11 | 12.3E-10 | 12 |
| 12 | 12e1000000000000000 | 12 |
| 13 | 122000e-2 | 1220 |
| 14 | +2147483649 | 2147483647 |

## 定义函数 itoa()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 输入（base=10 8 16） | 十进制 | 八进制 | 十六进制 |
| 1 | 123 | 123 | 0173 | 0X7B |
| 2 | -123 | -123 | 037777777605 | 0XFFFFFF85 |
| 3 | 100 | 100 | 0144 | 0X64 |
| 4 | -100 | -100 | 037777777634 | 0XFFFFFF9C |
| 5 | 2147483647 | 2147483647 | 017777777777 | 0X7FFFFFFF |
| 6 | -2147483647 | -2147483647 | 020000000001 | 0X80000001 |
| 7 | -2147483648 | -2147483648 | -2147483648 | 0X80000000 |

## 编译并运行第六章的桌面计算器程序

|  |  |  |
| --- | --- | --- |
| 序号 | 输入 | 输出 |
| 1 | 3 | output: 3 |
| 2 | age = 6 | output: 3 |
| 3 | age = 2;3\*(1+age+4\*3)\*5+2+2/2+pi | output: 2  output: 234.142 |
| 4 | 123+1222222\*55+45/9 | output: 6.72227e+07 |
| 5 | hahaha | output: 0 |
| 6 | (111-42\*4444+22)\*444; age=1212;pi+age+e\*222 | output: -8.28127e+07  output: 1212  output: 1818.38 |
| 7 | 666666666666666666666\*6666666666666 | output: 4.44444e+33 |

# 实验总结

* + - 1. 本次实验基本完成了题目要求的功能。
      2. 编写atoi函数和计算器这两题情况比较多，可能没有考虑到一些特殊情况。
      3. 编写atoi函数这题大量使用了string类型和string的相关函数，导致一些情况下转换速度会变慢。
      4. 使用了long long和int的转换，可能造成精度的损失。
      5. 很多地方使用了string和const char \*的转换，以及可能会造成未知错误。
      6. 写代码的时候有很多地方没有考虑清楚，导致调试花费的时间很长。