



|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | 17130130312 | **姓名** | 曹晨瑶 |
| **班级** | **1713013** | **任课教师** | **王献青** |
| **实验名称** | 函数编程 | | |
| **实验学期** | **2018 – 2019 学年第2学期** | | |
| **实验日期** | 2019年5月4日 | **实验地点** | **G334** |
| **报告成绩** |  |  |  |

西安电子科技大学软件学院

# 实验目的

本次实验通过设计实现几道以自定义函数为主的程序，以熟悉 C++为程序员提供的函数相关机制，加深对命令行参数、参数传递方式、函数形参的特殊形式等知识的感性认识。学会过程式程序设计范型的基本策略，能够灵活运用相应机制，提高学生的程序设计水平。

# 实验环境

操作系统：Windows10

开发工具: Code::Blocks

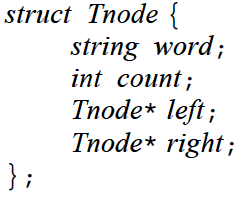
# 实验内容

## 命令行参数

Write a program like ‘‘Hello, world!’’ that takes a *name* as a command-line argument and writes ‘‘Hello, *name*!’’. Modify this program to take *any number of names* as arguments and to say hello to each.

## 二叉树及其操作

Consider:



Write a function for entering new words into a tree of *Tnode*s.

Write a function to write out a tree of *Tnode*s.

Write a function to write out a tree of *Tnode*s with the words in alphabetical order.

## 可变参数表

Write an *error* function that takes a *printf*-style format string containing %*s*, %*c*, and %*d* directives and an arbitrary number of arguments. Don’t use *printf*(). Look at §21.8 if you don’t know the meaning of %*s*, %*c*, and %*d*. Use <*cstdarg*>.

## 日期计算

Write functions to add one day, one month, and one year to a *Date* as defined in §5.9[13].  
Write a function that gives the day of the week for a given *Date*.

Write a function that gives the *Date* of the first Monday following a given *Date*.

# 数据结构与算法设计

## 命令行参数

**模块结构及文件组织设计：**

模块1：主控模块，仅包括文件main.cpp ，定义了 main()函数。

**关键数据结构设计：**

无

**算法1.1 int main()**

作 用：主控函数,通过命令行参数读取数据

参 数：int argc char \*argv[]

返回值：总是返回0。

计算过程：

1. 通过main()函数的参数读取命令行参数（都是字符串）；
2. 通过for循坏对获取的每个输入进行输出；
3. 返回0.

## 二叉树及其操作

**模块结构及文件组织设计：**

模块1：仅包括main.cpp文件,定义了main（）函数。

模块2：仅包含function.hpp文件，对regist\_word()、deleteBST()、search()、print\_tree()、print\_tree\_byorder()四个函数进行了声明。

模块3：仅包含function.cpp文件，对regist\_word()、deleteBST()、search()、print\_tree()、print\_tree\_byorder()四个函数进行了功能实现。

**关键数据结构设计：**

struct Tnode{

string word;

int count;

Tnode \*left;

Tnode \*right;

};

**算法2.1 int main()**

作 用: 主控函数 输入数据、初始化、输出结果

参 数： 无参数

返回值： 0

计算过程：

1. 建立根结点并对其初始化
2. 输入数据同时调用算法2.2；
3. 调用算法2.5；
4. 调用算法2.6；
5. 调用算法2.3；
6. 判断是否删除成功；
7. 返回0.

**算法2.2 Tnode\* regist\_word()**

作 用： 构造二叉排序树 递归调用

参 数： Tnode\* &root, const string& theword

返回值： root

计算过程：

1. 若二叉排序树树为空，则创建一个新结点作为根节点；
2. 将插入的数据与根结点的word比较，若相等，则根结点的count加1；
3. 若小于（首字母按照字母表顺序在根节点word的左边），则插入根结点的左子树中;
4. 若大于（首字母按照字母表顺序在根节点word的右边），则插入根结点得右子树中;
5. 返回root.

**算法2.3 Bool deleteBST()**

作用：销毁二叉树

参数：Tnode\* root

返回值：true或者 false

计算过程：

1. 若根结点为空（树为空）,返回true;
2. 递归调用deleteBST(root->left),删除左子树;
3. 递归调用deleteBST(root->right),删除右子树；
4. 删除root；
5. 返回true.

**算法2.4 bool search（）**

作用：查找数据是否存在

参数：Tnode\*root，const string& theword

返回值：true 或者 false

计算过程：

1. 若树不存在，返回false;
2. 若根结点的word与要查询的数据相同，返回true；
3. 若不相同，递归调用本函数在左子树中查找；
4. 若不相同，递归调用本函数在右子树中查找；
5. 未找到，返回false.

**算法2.5 void print\_tree()**

作用：打印二叉排序树，先序遍历输出，按结点所在层次向右缩进打印

参数：Tnode\* root,unsigned int indent =0

返回值：无返回值

计算过程：

1. 若树为空，则终止函数；
2. 打印若干空格，向右缩进；
3. 输出根结点；
4. 递归调用，打印左子树；
5. 若根结点只有左子树没有右子树，向右缩进打印<NULL>;
6. 递归调用，打印右子树.

**算法2.6 void print\_tree\_byorder()**

作用：打印二叉排序树，按照结点中的应用数据按照word的字典顺序输出（二叉排序树采用“中序遍历”）

参数：Tnode\* root

返回值：无返回值

计算过程：

1. 若树为空，则终止函数；
2. 递归调用，打印左子树；
3. 输出根结点；
4. 递归调用，打印右子树.

## 可变参数表

**模块结构及文件组织设计：**

模块1：包括文件main.cpp,定义了main（）函数和error()函数

**关键数据结构设计：**

无

**算法3.1 int main()**

作 用： 主控函数 初始化、函数调用和结果输出

参 数： 无参数

返回值： 返回0

计算过程：

1. 定义字符串指针char\* a 并初始化；
2. 定义字符char b并初始化；
3. 定义字符串指针char\* c并初始化；
4. 定义整型 d 并初始化；
5. 调用算法3.2
6. 返回0.

**算法3.2 void error()**

作 用：实现printf风格的错误报告函数

参 数： const char \* format…

返回值： 无

计算过程：

1. 函数的参数从右向左依次入栈
2. 定义字符指针ptr指向format所指的字符串
3. 定义ap指针指向arg1参数的起始地址
4. format指针对参数进行遍历，碰到“%X”（X为d、c、s、）则输出ap地址上所指向的变量的值，同时ap指向下一个参数的地址
5. 遍历结束，ap指针赋值为NULL

## 日期计算

**模块结构及文件组织设计：**

模块1：仅包含main.cpp文件，仅包含main（）函数

模块2: 包含function.hpp文件何function.cpp文件，包含关键数据结构和其他函数声明和定义

**关键数据结构设计：**

struct Date

{

int year;

int mon;

int day;

};

**算法4.1 int main（）**

作用：主控函数 输入数据、函数调用、输出结果

参数：int year

返回值：0

计算过程：

（1）定义Date 结构体类型的一个变量

（2）调用算法4.3并输出

（3）调用算法4.4并输出

（4）调用算法4.5并输出

（5）调用算法4.6并输出

（6）调用算法4.7并输出

（7）返回0

**算法4.2 bool isleap()**

作用：判断某一年是否是闰年

参数：int year

返回值：True/False

计算过程：

（1）判断year能否被400整除或者能被4整除而不能被100整除

（2）如果能，返回True；不能的话返回False

**算法4.3 Date add\_day()**

作用：计算对特定日期递增一天后得到的新日期

参数：Date date

返回值：Date newdate

计算过程：

（1）首先定义一个新的Date类型变量newdate

（2）将传入的date的年份和月份赋给newdate

（3）判断是否为闰年

（4）分别对闰年和非闰年情况下对不同长度的月份的边界情况进行处理，也就是处理对每月的最后一天递增一天后的特殊情况处理。

（5）如果是12月的最后一天再做进一步处理（年份加一，月份、日期置1）

（6）返回newdate

**算法4.4 Date add\_mon()**

作用：计算对特定日期递增一个月后得到的新日期

参数：Date date

返回值：Date newdate

计算过程：

（1）首先定义一个新的Date类型变量newdate

（2）将传入的date的年份、日期赋给newdate，月份加1赋给newdate

（3）判断是否为1月

（4）如果是1月的最后一天，要根据闰年非闰年，对日期赋不同的值（28或29）

（4）其他月份，按照31天和30天分为两类，对每个月的最后一天进行特殊处理（如果下一个月和自己长度不同，要给日期重新赋值）

（5）如果是12月再做进一步处理（月份赋值1，年份加一）

（6）返回newdate

**算法4.5 Date add\_year()**

作用：计算对特定日期递增一天后得到的新日期

参数：Date date

返回值：Date newdate

计算过程：

（1）首先定义一个新的Date类型变量newdate

（2）将传入的date的月份和日期赋给newdate，年份加一

（3）判断是否为闰年

（4）分别对闰年和非闰年情况下对2月最后一天进行特别处理（原来是29，现在应该是28）.

（5）返回newdate

**算法4.6 int \_day\_of\_week()**

作用：计算特定日期是星期几

参数：Date date

返回值：int day\_of\_w

计算过程：

（1）把2019.1.1作为基准，相差天数初始为0

（2）判断年份是否大于2019

（3）对于2019之前的年份，第一年需要单独判断，看是否为闰年以及月份是否在2月之前.2月之后的闰年当非闰年处理

（4）从该日期开始，年份依次递增到2018，闰年非闰年天数依次,减366或365

（5）从该日期开始月份依次递增到12月，天数依次递减该月天数

（6）最后减去32和当前date.day的差值得到和2019.1.1相差的天数（负值）。

（7）求余数得到是星期几并返回

（8）2019年及其之后的年份同之前，只是每次不在想减，而是加上对应天数.

**算法4.7 Date next\_monday()**

作用：求出指定日期之后的第一个星期一的日期

参数：Date date

返回值：Date newdtae

计算过程：

（1）获取传入的日期的星期几

（2）计算和下一个周一相差的天数

（3）将传入的日期加上求得的天数赋值给newdate

（4）判断日期是否超出了该月的天数，超出，则月份加一，天数取余

（5）判断月份是否溢出，超过12的话，月份对12取余，年份加一

（6）返回newdate

# 测试用例与测试结果

## 5.1 命令行参数

|  |  |  |
| --- | --- | --- |
| 序号 | 测试数据 | 输出结果 |
| 1 | Tom WER ll  “jk l” “kkk” | 结果是：  Hello Tom  Hello WER  Hello ll  Hello jk l  Hello kkk |

## 5.2 二叉树及其操作

|  |  |  |
| --- | --- | --- |
| 序号 | 测试数据 | 输出结果 |
| 1 | b  a  g  f  e  e  d  c | b: 1  a: 1  g: 1  f: 1  e: 2  d: 1  c: 1  <NULL>  <NULL>  <NULL>  <NULL>  a: 1  b: 1  c: 1  d: 1  e: 2  f: 1  g: 1  delete successfullly |

## 5.3可变参数表

|  |  |  |
| --- | --- | --- |
| 序号 | 测试数据 | 输出 |
| 1 |  | Hello, Rainman is a super man. 2016.  We are netter.  Hey, everyone. I am spiderman , I am 20 . |

## 5.4日期计算

|  |  |  |
| --- | --- | --- |
| 序号 | 测试数据 | 输出 |
| 1 | 2018 5 4 | add\_day:2018 5 5  add\_mon:2018 6 4  add\_year:2019 5 4  xingqiji:Thursday  next\_monday:2018 5 8 |
| 2 | 1955 2 28 | add\_day:1955 3 1  add\_mon:1955 3 28  add\_year:1956 2 29  xingqiji:Monday  next\_monday:1955 3 7 |
| 3 | 2020 12 31 | add\_day:2021 1 1  add\_mon:2021 1 31  add\_year:2021 12 31  xingqiji:Thursday  next\_monday:2021 1 4 |

# 实验总结

通过这次实验，我掌握了c++命令行参数的使用方法，采用模块化的方法进一步理解了“二叉树”这一数据结构，对函数编程中可变参数的应用有了初步的了解，最后又用c++对大一做过的日历计算进行了复习。第一题中，习惯了之前的标准化输入和输出，突然使用这种命令行模式下的输入，感觉对“编程语言”又有了新的认识。第二题中，由于上学期刚学了数据结构，使用二叉树这一典型的数据结构处理这一问题，还算比较顺利，比较意外的是之前不知道string类型支持字典序的直接比较，本来想着要转化一下，使用strcmp（）函数，后来直接str1和str2使用大于小于相等比较，简单了许多。第三题在解决过程中查阅了老师给的相关资料，又在网上搜索了其他相关资料，解决了问题，也给我的学习打开了一个新的方向，标准库的函数本身就是我们学习的一个好地方（之前一直忽略了，只知道用，却从来没有想过它是如何实现的）。最后的第四题，对于年份、月份的不平均、异常情况很多这一挑战，在不断Debug中，慢慢解决了一个又一个边界问题，对于逻辑处理比较多的时候，发现分成小问题不失为一个高效的处理方法。