|  |  |
| --- | --- |
|  |  |
| **学 号：** | 0121801100913 |

****

**《面向对象程序设计C》**

**课程大作业**

|  |  |
| --- | --- |
|  |  |
| **学 院** | 计算机科学与技术学院 |
| **专 业** | 软件工程 |
| **班 级** | 软件zy1802 |
| **姓 名** | 张淳慷 |
| **指导教师** | 王云华 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2020 | 年 | 6 | 月 | 3 | 日 |

|  |  |
| --- | --- |
| **选**  **题说明** | **题目:** **设计一个日期类，实现时间的加、减操作。** |
| **作**  **业 内**  **容** | 一、基本要求：  1. 描述设计和相关算法思路  2. 类体重要定义对日期的初始化构造函数，包含一个重载默认值方式；  3. 日期包括年月日，需要考虑日和月的进位对月年的改变问题，特别注意不同月份天数的问题，判断闰年问题；  4. 当做日期加或减操作时，需要计算和显示新的年月日。（此处要求用到+或-，++或--运算符重载，对日期对象进行加或减n天，加或减1天）；  提示：date& date::operator +(int d3)   {      this->day+=d3;  ……    return \*this;   }    date& date::operator ++()  {  this->day++;  ……  return \*this;  }  5. 自行设计测试用例观察结果，注意一些自认为比较重要或特殊数据节点的测试  6. 对本大作业进行总结（存在的不足、问题、经验等）  二、提高要求：  1. 通过继承方式设计出时间类（包含年月日时分秒），给出类的定义即可  2. 此时对时间加减需要考虑哪些问题？能否试图扩展想象一下设计一个完整的日历表的思路？  三、其他要求：  1. 独立完成，形成大作业报告（文档的页数可根据需要增减），包含设计思路、程序代码、测试结果和总结（提高要求部分可以融入其中），确实不能完整完成，可以完成其中一部分或者用另外的方法。  2. 公开答辩先由个人提出申请，抽时间组织在线公开答辩（答辩需要对大作业报告自述和回答老师和学生的提问），答辩成绩在期末最终成绩中加算，最多会另加5分（总分最多100分）  3. 被认定为抄袭或被抄袭者，本课程最终成绩作不及格处理。  4. 在网络教学平台上提交大作业报告和源代码（仅仅是代码），最后截止日期为：6月16日。  四、评分标准：   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 基本1 | 基本2 | 基本3 | 基本4 | 基本5 | 基本6 | 提高1 | 提高2 | 报告 | 公开答辩 | | 10分 | 5分 | 15分 | 15分 | 15分 | 10分 | 5分 | 5分 | 20分 | 另加 |  相关算法设计思路  * 1. **设计思路：**   不妨先分析题目所要求达到的基本功能：   1. 重载++，-，+，-符号，实现日期的加减。 2. 输入想要的日期以及对应的加减天数，输出对应的日期以及其距离，尤其是注意天数与月份年份之间的转化。 3. 距离另一天的日期天数之差 4. 判定闰年，并对之可以进行相应的年份转换 5. 两种构造函数，默认、有初始化成员变量的。   **1.2 算法思路：**  1）.正常对成员变量进行赋值，注意深拷贝问题  2）.重载+和-时需要尤其考虑日、月、年的进位关系以及不同月份下的日期数和闰年所对应的366的问题，为保证美观和使用方便，可将判定闰年的代码进行封装，同时也可对应平年和闰年建立相关联的月份数组，如下：  const int large\_ = 366;  const int small\_ = 365;  const int Month\_small[13] =  { 0,31,28,31,30,31,30,31,31,30,31,30,31 };  const int Month\_large[13] =  { 0,31,29,31,30,31,30,31,31,30,31,30,31 };  通过判断年份能否被400整除或者是被4整除而不被100整除得出该年份是否为闰年，并返回相应的true和false。  **构造函数：**  Date::Date()  {  this->tday = 0;  this->tmonth = 0;  this->tyear = 0;  this->tsum = 0;  }//初始化  Date::Date(int year, int month, int day)  {  this->tyear = year;  this->tmonth = month;  this->tday = day;  this->tsum = 0;  if (large\_year(this->tyear))  {  for (int i = 1; i < month; i++)  this->tsum += Month\_large[i];  this->tsum += day;  }  else  {  for (int i = 1; i < month; i++)  this->tsum += Month\_small[i];  this->tsum += day;  }//得到距离总天数  }  **1.3核心代码实现以及思路讲解**  **（以下代码为保证可读性，只展示判断为闰年的情况，平年情况仅有数组调用的差别，判定闰年函数已在上方给出）（large\_year即为判定闰年）**  1)重载+号：  Date& Date::operator+(int number)  {  int year = this->tyear;  int sum = this->tsum;  sum += number;//得到加上日期之后的总天数  int i = 1;  7 if (large\_year(this->tyear))  {  if (sum >= large\_)  {  int j = sum / large\_;//判断年数  for (int i = 0; i < j; i++)  {  if (large\_year(year + i))  sum -= large\_;  else  sum -= small\_;  }  year += j;  }  21 }  在代码实现过程中不难发现，+号的重载核心在于需要判断加入后的总天数是否大于一年并根据其实闰年与否进行相应的总天数上的加减（上方所示代码的7至21行），我们通过前期封装好的large\_year进行判断，并通过除以当年的总天数以得出增加的年数，后进行根据所加上之后的年是否是闰年进行相应的总天数减法直到总天数少于366天时进行月份与日数的确定  if (large\_year(year))//判断月与日  {  while (sum > 0)  sum -= Month\_large[i++];  i--;  Date temp(year, i, sum + Month\_large[i]);  return temp;  }  在上方代码中发现，将剩余的总天数循环减去对应月份的天数直到其小于0时，不难想象出该日期所处的真正的月份应该处于上一个月份，而日期恰是该月份的总天数加上那个小于0的sum得到。于是返回加入对应天数后的结果。图式思路如下    2）重载减号 ：  重点在于如果所减去的日期数大于日期数的情况，小于日期数的情况只需重新将总天数赋值即可，**下方为大于总天数的情况**：  Date& Date::operator-(int number)  {  int year = this->tyear;  int sum = this->tsum;  int True\_sum = number - sum  int i = 1, K = True\_sum / small\_;  for (int i = 1; i <= K; i++)  if (large\_year(year - i))  {  True\_sum -= large\_;  }  else  {  True\_sum -= small\_;  }  通过将实际减少的天数（也可理解为总天数为负数）与年的的商为大致的年数（可能有闰年，所以仅仅只为大致的），  并将其减少对应的天数  year -= K;得到对应的年数  if (tsum <= number)  {  year--;  if (large\_year(year))  sum = large\_ - True\_sum;  else  sum = small\_ - True\_sum;  }  因为在所得的年数上，总天数依然小于0，所以结果所处的年应在该年的前一年，并可以此得到该日期在该年的总天数  if (large\_year(year))//判断月数  {  while (sum > 0)  sum -= Month\_large[i++];  i--;  Date temp(year, i, sum + Month\_large[i]);  return temp;  }  }    3）相差天数的实现  int Date::operator-(Date& date)  {  int sum\_date = 0;  int i = 0, j = abs(date.tyear - this->tyear);//判断年差  int year = date.tyear;  if (large\_year(date.tyear))  {  if (j==0) {  sum\_date += (this->tsum - date.tsum);//判断假设地址恰相隔一年下时的日期差  }  else {  sum\_date += (this->tsum - date.tsum+large\_);  }  }  通过判断年差是否为0来进行相差天数的计算，其思路可由下图表示    for (i = 1; i < j; i++)  {  if (large\_year(year + i))//根据年差换算成日期差  sum\_date += large\_;  else  sum\_date += small\_;    return sum\_date;  } 测试用例     调试分析：通过上图可以看出，在改程序已经可以成功地解决闰年与平年直接的跨越以及2.29在闰平年之间的转换问题，而通过第四张图片不难发现对于随意的天数的加减，改程序也可以正确的得出答案，所以受该程序可以正常执行。   提高  * 1. 含有时分秒类的定义：   class time :public Date {  int thour, tmin, tsec;  time(int year, int month, int day, int hour, int min, int sec) :Date(year, month, day) {  thour = hour;  tmin = min;  tsec = sec;  }};  3.2此时需要考虑的问题以及项目展望：  不难看出，此时需要注意的就是时分秒以及日之间的进位问题进行相应的重载比如60秒进为1分，60分进为1时，24时进为一天，根据在本学期所学到的软件工程基础的知识我们可以想到建立一个完整的包含年月日时分秒的电子时钟，采用可视化程序设计，建立一个包含视图类、文档类对话框类等的工程，  其主要核心功能，即时间的加减可放于视图类中，以视图类为例，其类图如下    Runtime（）用于计算时间，Week用于计算星期、Change用于将int型转化为CString型、GetDateTIme用于获取系统时间、Onreset（）用于重新获取系统时间，Ondraw函数用来对时间日期进行输出以及计时器消息。  Runtime（）用于计算时间时，可以如此进行，从系统时间获取秒，每次达到一秒的时候就对程序中的second进行加1，达到60秒时就对minute加1，以此类推，其中关于闰年的问题已在本次课设中得到解决就不过多赘述，最终可以得到正确的时间。  其中，Ondraw（）是核心内容，我们可以采用内存绘图的方法，在内存中创建一个与屏幕绘图区域一致的对象，使得重绘图像时大大提高运行速度。为了进一步保证计时的精确性，我们还可以选用一种获取系统两秒间隔作为计时间隔的方法以使得二者时间间隔一致，即通过Onreset（）在每次重绘视图的时候都重新获取系统时间，如果两次获得的秒不一致时，便对时间加一秒并通过一个计时器消息，使视图50ms刷新一次，最终保证时间的准确性。 4大作业感悟与收获 说实话作为一个转专业过来却又是18级的学生，自己的编码能力真的在这次大作业中得到了很大的锻炼，因为要补课的原因，自己并没有很多时间进行大量的编程练习，做过的大型项目可能仅限于学校的大作业，于是乎有的时候我甚至会觉得自己的编码能力并比不上一个比自己低一年的计算机在读的学弟学妹，但是比他们强的地方又在于我已经系统的学过了数据结构与算法等等的知识，所以说不得不说我这种情况应该算是学生中比较少见的理论能力超强（毕竟上两门都94+），然而实践能力还有待提高，这次c++的学习其实就给了我很大很大的实践机会，让自己既学到了面向对象编程知识的同时又让自己的编程能力得到了锻炼。  就拿自己这次的大作业为例子，其实一开始抱着理论的角度，这次大作业根本不存在说有算法上的难度，基本上就是一个平铺直叙的过程，然而我在实际编码的时候确实碰到了不小的问题，因为我是分两天写完这个代码的，而之前最开始的时候我并没有对于月份以及年份等进行相应的const变化，也就导致我每一次都是直接输入的数字，老实讲如果是自己一个人一口气完成而且不存在bug的情况下，这种方式可能并没有太多的问题，然而我在后来大概3到4天的时候重新打开写程序时，明显感觉到自己的程序十分的凌乱而且有的时候数字的众多导致我没有办法正常的读完自己的代码代入带当时的我的思路中，只能收一点一点的看，十分浪费时间，于是我干脆推到全部重来，便有了现在我这个看起来比较美观的代码，把月份的日期都封装到了数组里面，然后判断天数是只需要直接对数组进行遍历的加或者减便可以了。  还有令我影响很深的一点是关于闰年闰月问题，其实本来我觉得将其化为366天的时候就可以直接忽略这个问题了，然而实际上如果不加入特别的判断语句进行判定，2000.2.29在加入365天之后会变为2001.2.29，一个根本不存在的日期，当时刚开始调试时碰到这个问题时，我返回去看自己的代码都不知道从何改起，还好之后想到可以利用闰年的判定函数规避这个错误才最终迎刃而解，得到了正确的答案。  其实我一直是一个很喜欢考试的人，尤其是理科类的考试，因为他们只有对错，不存在说有老师主观打分的问题，但是因为疫情的到来，许多课程都改为了大作业的形式，也导致了我这一学期代码量的迅速上升，让我实战能力得到了锻炼，对于自己代码能力的上升我实在是再开心不过了，因为这意味着我离自己的梦想学校——帝国理工的又一次靠近，希望在接下来的日子中，我可以继续锻炼自己的理论高度与代码能力，最终到达我理想的院校！ |
| **成绩评定** | **教师签名：** |