2020 年 OOP 上机实验(2)

(Ver. 0.67 2020/03/16 wangxp@fudan.edu.cn)

一、实验目标

1）理解函数重载的定义以及规则。函数名可以重载（override）:同一个函数名，可以定义多个函数——只要这些函数在参数的个数或者类型上有差异就行了。

2）测量程序运行的时间；

3） 理解函数定义使用不同形式的参数的导致的性能差异。

二、实验内容

1.1 第 1 部分：函数的重载（参见书 76，参见 msdn）

系统在调用重载函数时，按照一定的规则匹配合适的函数：1）找到完全匹配的函数；2）已执行不重要的转换；3）已执行整型提升；4）已存在到所需参数的标准转换；5）已找到省略号表示的参数。

实验过程：

把 E2_11.cpp 放在项目工程中，调试程序，并回答如下问题：

1）函数重载的参数匹配和转换，有哪些规则？尝试转换的顺序是怎样的？

2）编译 E2_11.cpp 代码，说明哪一个调用没有匹配的函数？为什么？

3）为上述代码补写一个重载函数(返回值为（a+b）/2，返回值类型为 float)；写出成功编译后，程序运行的结果。

4）如果此时把 E2_11.cpp 中原先定义的两个 reckon 函数都删除(注释),再运行的结果是怎样的？为什么？

1.1 提交要求：回答上述问题并提交第 3）问修改好的代码。

1.2 第 2 部分：测量程序的运行时间

第 2 部分的实验，要求禁止编译优化：原因是优化后的代码，与 C++语言代码有很大的差异。**对于 makefile，只需修改第 23 行，将 CFLAGS 中的参数-o3 修改为-o0**。使用其他编译器的，请自行查找如何禁止编译器优化。

实验过程：

下载 stopwatch.h，stopwatch.cpp，E2_12.cpp 代码，放入一个工程中，编译运行。仔细阅读代码，记录不同 N 值(100、1000、10000、100000)时程序执行时间，每个 N 值运行 5 次，填写下表：

N 取值	100	1000	10000	100000	预测 100000000
第 1 次					预测时间 ：(us 毫秒)
第 2 次					
第 3 次					
第 4 次					预测理由：
第 5 次					
平均时间					

1.2 提交要求：不需要修改源代码，也不需要上传源代码文件；但是需要填写上表。

1.3 第 3 部分：函数定义使用不同形式的参数导致的性能差异

第 3 部分的实验，同样要求禁止编译优化。

实验过程：

1）下载 1MI.txt 文件(内含 1 百万个整数，分布范围在 1~10^{10}之间)。首先生成 2MI.txt、4MI.txt 和 8MI.txt：把提供的 1MI.txt 经过多次拷贝即可。

2）E2_13.cpp 已经完成了程序的框架，但是 4 个重要的统计函数没有实现，请在 **TODO 1、TODO 2、TODO 3 和 TODO 4** 行补全。这 4 个统计函数分别申明为

a) (b.1) double reckon_avg(const vector<int>& vec);
   (b.2) double reckon_avg_2(vector<int> vec);

b) (c.1) int statistic_avg(const vector<int>& vec, double avg);
   (c.2) int statistic_avg_2(vector<int> vec, double avg);

其中 reckon_avg*计算均值；而 statistic_avg*是求接近于均值的整数的个数。这两组函数功能相同，但是使用了不同的参数形式。

3）统计 1M、2M、4M 到 8M 个整数的均值，并计算最接近均值的整数（对均值四舍五入得到的整数）的个数。

4）请在下表中填入输入 8M 个整数时的各个环节的时间值及均值。

代码	IO时间	b.1	c.1	b.2	c.2
第 1 次					
第 2 次					
第 3 次					
第 4 次					
第 5 次					
平　均					
最优方法(打勾)					

1.2 提交要求：提交修改后的 E2_13 源代码；同时还需要填写上表。

[注意] 我们没有介绍文件操作，可通过控制台的 IO 重定向解决文件操作的问题，参考 csdn。

2020 年 OOP 上机实验(2)

提交检查表

1.1 回答下列问题并提交源代码 E2_11.cpp。

1) 函数重载的参数匹配和转换，有哪些规则？尝试转换的顺序是怎样的？

规则：使用重载函数时应该与该函数的参数类型和数量一一对应；若主函数中重载函数的参数类型无与之匹配的函数，则参数会按照一定的转换顺序升级（如果自己有定义的话会按照设置好的转换规则进行转换）；尽量不要定义相互转换的类（容易产生二义性）；尽量避免到内置算术类型的转换。

转换的顺序：char->int->long->float->double

2) 编译 E2_11.cpp 代码，说明哪一个调用没有匹配的函数？为什么？

`reckon(3.0f,4.0f)` 没有匹配的函数，因为它的两个实参都是 `float`。

3）为上述代码补写一个重载函数(返回值为（a+b）/2，返回值类型为 float)；写出成功编译后，程序运行的结果。

```
reckon @ line 18
2.33333
reckon @ line 13
1.75
reckon @ line 23
3.5
```

4）如果此时把 E2_ov.cpp 中原先定义的两个 reckon 函数都删除(注释),再运行的结果是怎样的？为什么？

结果如下：

```
reckon @ line 13
3.5
reckon @ line 13
3.5
reckon @ line 13
3.5
```

原因：参数类型自动转化，比如 `reckon(3.0f,4)`的第二个参数是 `int` 类型，而 `reckon` 函数第第二个参数是 `float` 类型，所以根据转换规则，可以调用 `reckon` 函数并且把 4 转成了 `float` 类型，然后代数函数计算。

1.2 填写下表，不提交源代码

N 取值	100	1000	10000	100000	预测　　100000000
第 1 次	0ms	2.028ms	241.825ms	26315.9ms	预测时间 ：(ms 毫秒) 32809369472 预测理由： 有一定的线性规律，每增加一个 0 位数增加 2 位，具体数字稍微拟合以下。
第 2 次	0ms	2.033ms	246.34ms	27542.1ms	
第 3 次	0ms	2.958ms	242.352ms	26182.6ms	
第 4 次	0ms	2.026ms	241.353ms	25650.6ms	
第 5 次	0ms	1.963ms	244.247ms	26641.1ms	
平均时间	0ms	2.202ms	243.223ms	26466.5ms	

1.3 填写下表：**选择 8M 个文件**

代码	IO 时间	b.1	c.1	b.2	c.2
第 1 次	38640.081us	66.82200us	97.01700us	70.01100us	127.05800us
第 2 次		72.80600us	97.77400us	70.78300us	142.61000us
第 3 次		76.78500us	101.74200us	82.73900us	136.23500us
第 4 次		70.34800us	99.80800us	73.80800us	133.34100us
第 5 次		68.60800us	99.73200us	71.85800us	138.96300us
平　均		71.07380us	99.21460us	73.83980us	135.64140us
最优方法 (打勾)		√	√		

同时提交修改后的源文件 E2_13.cpp。