

重 庆 大 学

学 生 实 验 报 告

实验课程名称 计算机组成与结构

开课实验室 DS1501

学 院 软件学院 年级 2018 专业班 3(交换生) 班

学 生 姓 名 都景会 学 号 L1800025

开 课 时 间 2018 至 2019 学 年 第 1 学 期

总 成 绩	
教师签名	

软件学院制

《计算机组成与结构》实验报告

开课实验室：DS1501

2018 年 10 月 30 日

学院	软件学院	年级、专业、班	交换生(大 3),大 数据和软件工 程	姓名	都景会	成绩	
课程 名称	计算机组成与结构	实验项目 名 称	实验 3,4,5		指导教师	熊敏	
教 师 评 语	<div>教师签名：</div> <div>年 月 日</div>						

一、实验目的

实验 3: 深入理解 C++程序内存布局

实验 4: 程序性能分析

实验 5: C++代码性能优化

二、实验原理

实验三 C++程序内存布局, Intel inspector内存错误检测工具

Using Intel inspector to analyse memory leak, unsafe memory access etc. Intel inspector is a great tool to improve software stability.

实验四 程序执行时间测量, Intel Amplifier性能分析器

Using Intel Amplifier to analyse and compare efficiency of programs. Intel Amplifier is a great tool to compare time/memory complexity.

实验五 条件分支预测, 循环性能优化, 空间换时间, Cache性能优化, 使用Intel C++ Compiler

Understand ways to improve program performance.

三、使用仪器、材料

VC++, Intel Inspector, Intel Amplifier.

四、实验步骤

实验 3:

3.1 请先人工分析下面一段代码中存在的内存错误，然后再用 Intel inspector 进行内存检查，试分析检查结果。

```
代码: int arrayA = {0};  
  
cout<<arrayA[10]<<endl;  
  
return 0;
```

3.2 请任写一段包含内存错误的代码，然后交给 Intel inspector 来检查，分析检查结果。

3.3 在你的编程经历中，你曾经遇到过哪些难忘的与内存错误有关的 bug，请回忆你当时场景和你的调试经过，并思考，在今后的编程实践中，如果利用 Intel inspector 来增强对 C++代码中内存错误的调试能力，请写下你的体会，字数不限。

实验4:

4-1 请任意实现一种数值积分算法, 计算函数 $f(x) = e^x + \lg(\sqrt{x^2 + (1-x)^2})$ 在区间 $[0, 10]$ 内定积分 $\int_0^{10} f(x)$, 用本章学习的程序性能分析方法给出程序性能分析结果。

提示: 数值积分算法在《数值分析》这门课程中讲授, 常见的数值积分算法有梯形公式, 辛普森公式等。

实验5:

5.1 请用本节学过的代码优化知识优化你前面写过的数值积分程序, 并比较优化后的程序和优化前的程序的性能差异

五、实验过程原始记录(数据、图表、计算等)

3.1

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int arrayA = { 0 };
6      cout << arrayA[10] << endl;
7      return 0;
8  }
```

expression must have pointer-to-object type

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int arrayA[10] = { 0 };
6      cout << arrayA[10] << endl;
7      return 0;
8  }
```

C:\WINDOWS\system32\cmd.exe
-858993460
Press any key to continue . . .

Locate Memory Problems

Target Analysis Type Collection Log Summary

Problems

No Problems Detected

Intel Inspector detected no problems at this analysis scope. If this result is unexpected, try rerunning the target using an analysis type with a wider scope. Press F1 for more information.

3.2

```
1 #include <iostream>
2
3 int main()
4 {
5     using std::cout;
6     using std::endl;
7
8     int* a = new int;
9     *a = 5;
10    int *b(a);
11
12    cout << *a << ' ' << *b << endl;
13    delete a;
14    cout << *a << ' ' << *b << endl;
15    return 0;
16 }
17
```

C:\Users\user\source\repos\Project23\Debug\Project23.exe

5 5
-572662307 -572662307
Press any key to continue . . .

Locate Memory Problems

Target Analysis Type Collection Log Summary

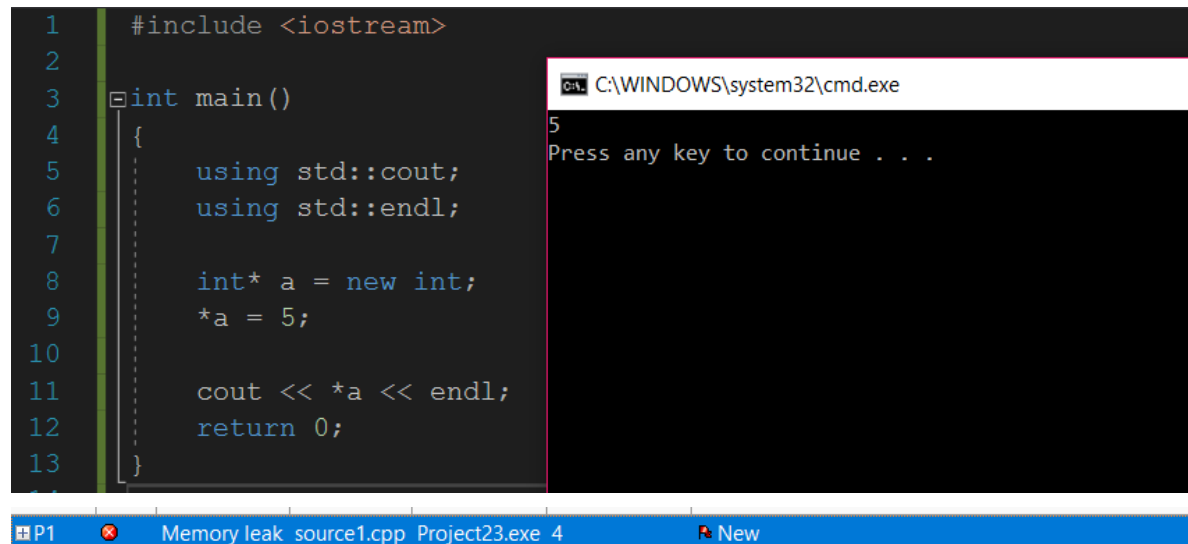
Problems

ID	Type	Sources	Modules	Object Size	State
P1	Invalid memory access	source1.cpp	Project23.exe		New

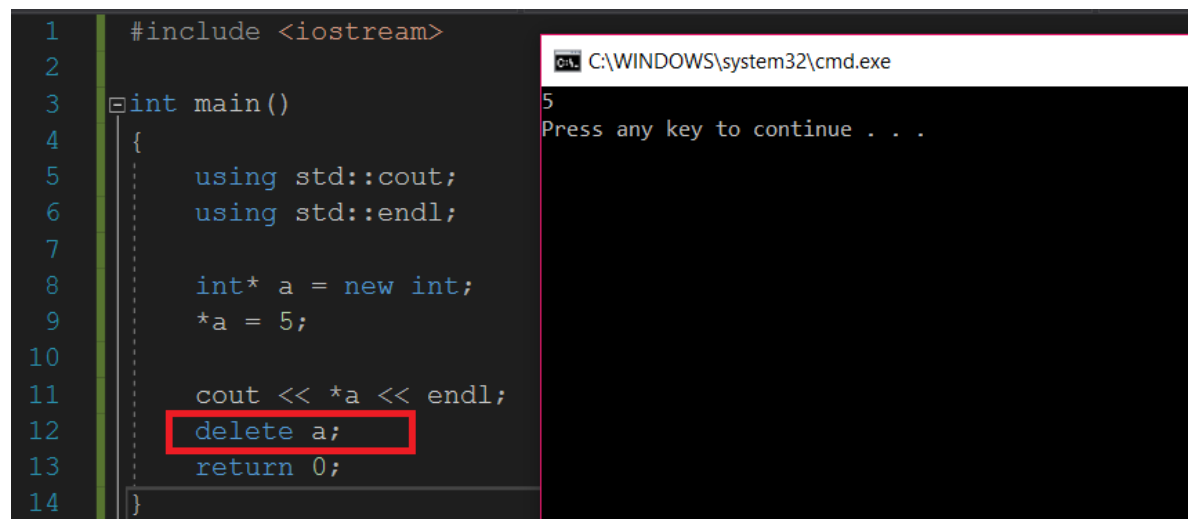
Code Locations: Invalid memory access

Description	Source	Function	Module	Object Size	Offset	Variable
Read	source1.cpp:14	main	Project23.exe		0x01572df8	
12	cout << *a << ' ' << *b << endl;				Project23.exe!main - source1.cpp:1	
13	delete a;				Project23.exe!0x12d99	
14	cout << *a << ' ' << *b << endl;				Project23.exe!0x12c0b	
15	return 0;				Project23.exe!0x12aa8	
16	}				Project23.exe!mainCRTStartup - exe	

3.3



Code Locations: Memory leak						
Description	Source	Function	Module	Object Size	Offset	Variable
Allocation site	source1.cpp:8	main	Project23.exe 4	block allocated at source1.cpp:8		
6	using std::endl;					Project23.exe!main - source1.cpp:8
7						Project23.exe!0x12d99
8	int* a = new int;					Project23.exe!0x12c0b
9	*a = 5;					Project23.exe!0x12aa8
10						Project23.exe!mainCRTStartup - exe



No Problems Detected

Intel Inspector detected no problems at this analysis scope. If this result is unexpected, try rerunning the target using an analysis type with a wider scope. Press F1 for more information.

4-1.

```

1  #include <iostream>
2  #include <cmath>
3  #include <iomanip>
4
5  using namespace std;
6
7  //initializing e as const
8  const float e = 2.71828182845904523536;
9
10 float formula(float const x) {
11     float fm(0);
12     fm = pow(e,x)+log(pow((x * x)+(x-1),1/2));
13     return fm;
14 }
15
16 float trapezoidal(int startNo, int endNo) {
17     float answer(0);
18     for (int i = startNo; i < endNo; i++)
19         answer = answer + (formula(i) + formula(i + 1)) / 2;
20     return answer;
21 }
22
23 int main() {
24     cout << trapezoidal(0, 10) << endl;
25 }

```



Hotspots
Hotspots by CPU Utilization

INTELVTUNE AMPLIFIER 2

Analysis Configuration
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform

Elapsed Time ⓘ: **0.286s**

No data to show. The collected data is not sufficient.

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

No data to show. The collected data is not sufficient.

Hotspots Insights

If you see significant hotspots in the Top Hotspots list, switch to the **Bottom-up** view for in-depth analysis per function. Otherwise, use the **Caller/Callee** view to track critical paths for these hotspots.

Explore Additional Insights

Microarchitecture Usage ⓘ: **41.1%**

Use ⓘ **Microarchitecture Exploration** to explore how efficiently your application runs on the used hardware.

Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

Application Command Line: C:\Users\user\source\repos\Project24\Debug\Project24.exe

Environment Variables:
PATH=C:\Program Files (x86)\IntelSWTools\compilers_and_libraries_2019\windows\redist\ia32\compiler;%PATH%;
PATH=C:\Program Files (x86)\IntelSWTools\compilers_and_libraries_2019\windows\redist\ia32\compiler;%PATH%

Operating System: Microsoft Windows 10

Computer Name: DESKTOP-GKNFUDE

Result Size: 2 MB

Collection start time: 07:48:05 30/10/2018 UTC

Collection stop time: 07:48:06 30/10/2018 UTC

Collector Type: Event-based counting driver, User-mode sampling and tracing

Finalization mode: Fast. If the number of collected samples exceeds the threshold, this mode limits the number of processed samples to speed up post-processing.


```

1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  /*initializing e as const */const float e = 2.71828182845904523536;
5
6  float formula(float const x) {
7      float fm = 0;
8      fm = pow(e, x) + log(pow((x * x) + (x - 1)*(x - 1), 1 / 2));
9      return fm;
10 }
11
12 float simpson(int startNo, int endNo) {
13     float answer(0); float divNum(0);
14     divNum = (endNo - startNo) / float(10);
15     for (int i = startNo; i <= endNo; i++){
16         if (i == startNo || i == endNo)
17             answer = answer + abs((divNum)*(formula(i)));
18         else if (i % 2 == 1)
19             answer = answer + 4 * (abs((divNum)*(formula(i))));
20         else
21             answer = answer + 2 * (abs((divNum)*(formula(i))));
22     }
23     answer = float(answer/3);
24     return answer;
25 }
26
27 int main() {
28     cout << simpson(0, 10) << endl;
29     return 0;
30 }

```

```

C:\WINDOWS\system32\cmd.exe
22134.6
Press any key to continue . . .

```

Hotspots Hotspots by CPU Utilization

Analysis Configuration Collection Log Summary Bottom-up Caller/Callee Top-down Tree Platform

Elapsed Time: 0.312s

No data to show. The collected data is not sufficient.

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

No data to show. The collected data is not sufficient.

Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

Application Command Line: C:\Users\user\source\repos\Project24\Debug\Project24.exe
 Environment Variables: PATH=C:\Program Files (x86)\IntelSWTools\compilers_and_libraries_2019\windows\redist\ia32\compiler;%PATH%; PATH=C:\Program Files (x86)\IntelSWTools\compilers_and_libraries_2019\windows\redist\ia32\compiler;%PATH%
 Operating System: Microsoft Windows 10
 Computer Name: DESKTOP-GKNFUDE
 Result Size: 2 MB
 Collection start time: 07:46:07 30/10/2018 UTC
 Collection stop time: 07:46:08 30/10/2018 UTC
 Collector Type: Event-based counting driver, User-mode sampling and tracing

Finalization mode: Fast. If the number of collected samples exceeds the threshold, this mode limits the number of processed samples to speed up post-processing.

CPU

INTEL VTUNE AMPLIFIER 2019

Hotspots Insights

If you see significant hotspots in the Top Hotspots list, switch to the [Bottom-up](#) view for in-depth analysis per function. Otherwise, use the [Caller/Callee](#) view to track critical paths for these hotspots.

Explore Additional Insights

Microarchitecture Usage: 29.6%
 Use [Microarchitecture Exploration](#) to explore how efficiently your application runs on the used hardware.

5-1

```

1  #include <iostream>
2  #include <cmath>
3  #include <iomanip>
4
5  using namespace std;
6
7  //initializing e as const
8  const float e = 2.71828182845904523536;
9
10 float formula(float const x) {
11     float fm(0);
12     fm = pow(e, x) + log(pow((x * x) + (x - 1)*(x - 1), 1/2));
13     return fm;
14 }
15
16 float trapezoidal(int startNo, int endNo) {
17     float answer(0);
18     for (int i = startNo; i < endNo; i++)
19         answer = answer + (abs((formula(i) + formula(i + 1))) / float(2));
20     return answer;
21 }
22
23 int main() {
24     cout << trapezoidal(0, 10) << endl;
25 }

```

C:\WINDOWS\system32\cmd.exe

23831

Press any key to continue . . .

Hotspots
Hotspots by CPU Utilization

INTEL VTUNE AMPLIFIER 2018

Analysis Configuration
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform

Elapsed Time [?]: 0.292s

No data to show. The collected data is not sufficient.

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

No data to show. The collected data is not sufficient.

Hotspots Insights

If you see significant hotspots in the Top Hotspots list, switch to the [Bottom-up](#) view for in-depth analysis per function. Otherwise, use the [Caller/Callee](#) view to track critical paths for these hotspots.

Explore Additional Insights

Microarchitecture Usage : 34.9%

Use [Microarchitecture Exploration](#) to explore how efficiently your application runs on the used hardware.

Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

Application Command Line: C:\Users\user\source\repos\Project24\Debug\Project24.exe

Environment Variables: PATH=C:\Program Files (x86)\IntelSWTools\compilers_and_libraries_2019\windows\redist\ia32\compiler;%PATH%; PATH=C:\Program Files (x86)\IntelSWTools\compilers_and_libraries_2019\windows\redist\ia32\compiler;%PATH%

Operating System: Microsoft Windows 10

Computer Name: DESKTOP-GKNFUDE

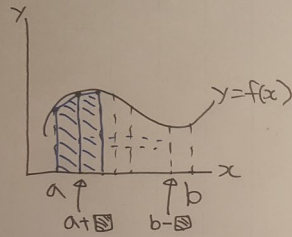
Result Size: 2 MB

Collection start time: 10:12:01 30/10/2018 UTC

Collection stop time: 10:12:02 30/10/2018 UTC

Collector Type: Event-based counting driver, User-mode sampling and tracing

Finalization mode: Fast. If the number of collected samples exceeds the threshold, this mode limits the number of processed samples to speed up post-processing.



if there are n steps to b (from a),

$$a + n \cdot \Delta x = b$$

$$f(a) \Delta x \text{ Area} = \frac{1}{2} (f(a) + f(a + \Delta x)) \cdot \Delta x$$

(梯形公式)

Approx Area by Trapezoidal rule

$$\int_a^b f(x) dx \approx \frac{\Delta x}{2} (f(a) + f(a + \Delta x) + f(a + 2\Delta x) + \dots + f(a + (n-1)\Delta x) + f(a + n\Delta x))$$

$$\approx \frac{\Delta x}{2} (f(a) + 2f(a + \Delta x) + \dots + 2f(a + (n-1)\Delta x) + f(b))$$

$$\approx \frac{\Delta x}{2} (f(a) + f(a + n\Delta x)) + \Delta x (f(a + \Delta x) + \dots + f(a + (n-1)\Delta x))$$

```


1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  //initializing e as const
7  const float e = 2.71828182845904523536;
8
9  float formula(float const x) {
10     float fm(0);
11     fm = pow(e, x) + log(pow((x * x) + (x - 1)*(x - 1), 1 / 2));
12     return fm;
13 }
14
15 float trapezoidal2(int startNo, int endNo) {
16     float answer(0);
17     answer = answer + (abs((formula(startNo) + formula(endNo) / float(2))));
18     for (int i = startNo+1; i < endNo; i++)
19         answer = answer + (abs(formula(i)));
20     return answer;
21 }
22
23 int main() {
24     cout << trapezoidal2(0, 10) << endl;
25 }
26

```

C:\WINDOWS\system32\cmd.exe

23831.5

Press any key to continue . . .

HotspotsHotspots by CPU Utilization

Analysis ConfigurationCollection LogSummaryBottom-upCaller/CalleeTop-down TreePlatform

Elapsed Time

0.271s

No data to show. The collected data is not sufficient.

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

No data to show. The collected data is not sufficient.

Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

Application Command Line: C:\Users\user\source\repos\Project24\Debug\Project24.exe

Environment Variables: PATH=C:\Program Files (x86)\IntelSWTools\compilers_and_libraries_2019\windows\redist\ia32\compiler;%PATH%; PATH=C:\Program Files (x86)\IntelSWTools\compilers_and_libraries_2019\windows\redist\ia32\compiler;%PATH%

Operating System: Microsoft Windows 10

Computer Name: DESKTOP-GKNFUDE

Result Size: 2 MB

Collection start time: 10:13:40 30/10/2018 UTC

Collection stop time: 10:13:42 30/10/2018 UTC

Collector Type: Event-based counting driver, User-mode sampling and tracing

Finalization mode: Fast. If the number of collected samples exceeds the threshold, this mode limits the number of processed samples to speed up post-processing.

Hotspots Insights

If you see significant hotspots in the Top Hotspots list, switch to the [Bottom-up](#) view for in-depth analysis per function. Otherwise, use the [Caller/Callee](#) view to track critical paths for these hotspots.

Explore Additional Insights

Microarchitecture Usage : 31.7%

Use [Microarchitecture Exploration](#) to explore how efficiently your application runs on the used hardware.

六、实验结果及分析

3.1

```
C:\WINDOWS\system32\cmd.exe
-858993460
Press any key to continue . . .
```

No Problems Detected

Intel Inspector detected no problems at this analysis scope. If this result is unexpected, try rerunning the target using an analysis type with a wider scope. Press F1 for more information.

习题内的代码有问题。错误的原因是代码没安排了 `array[10]` 的空间,然后用了 `cout` 函数。This problem can be solved by allocating “arrayA” with at least 10 slots. 在这报告我安排了 10 个空间,然后用了 intel inspector,确认了修改以后的代码没有问题

*However, since the value for `arrayA[10]` has not been allocated, it is set to trash value.

3.2

```
C:\Users\user\source\repos\Project23\Debug\Project23.exe
5 5
-572662307 -572662307
Press any key to continue . . .
```

Problems

ID ▲	Type	Sources	Modules	Object Size	State
P1	Invalid memory access	source1.cpp	Project23.exe		New

‘a’和‘b’都用同样空间,因为‘b’用 both use the same memory, which is designed by “shallow copy”. So after ‘a’ memory disappears, it is not possible to access memory where ‘b’ was pointing. Therefore, Intel inspector shows “invalid memory access” error message.

3.3

```
delete a;
```

No Problems Detected

Intel Inspector detected no problems at this analysis scope. If this result is unexpected, try rerunning the target using an analysis type with a wider scope. Press F1 for more information.

Memory leak 发现因为动态配额的存储空间没删除了. ‘a’存在 heap 存储器里面,所以 Intel inspector 可以检测. 在这报告,使用以后我删除了‘a’. 然后确认了代码没有 memory leak 问题.

4.1

Hotspots Hotspots by CPU Utilization

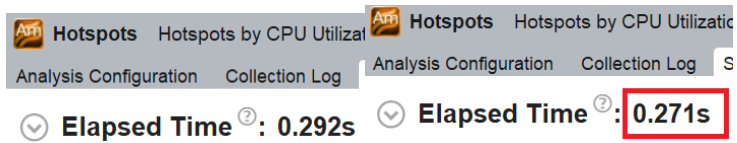
Analysis Configuration Collection Log S Analysis Configuration Collection Log S

Elapsed Time ②: 0.286s Elapsed Time ②: 0.312s

在这报告我用了 2 种算法;1)Trapezoidal rule; 2)Simpson’s rule. 因为 Simpson’s rule 需要时间分别称什么数(1/2/4), Simpson’s rule 使用时间比较长. 根据 Intel amplifier, 大概 0.035 秒慢.

Simpson’s rule

5.1



在这个习题用了 Trapezoidal rule. As we can see from the hand-written paper above, Trapezoidal rule can be simplified. Since Trapezoidal rule calculate $\{1/2(a+\square)\dots 1/2(a+(n-1)*\square)\}$ twice, simplifying the formula as the “Trapezoidal2(above)” reduces the time consumed by calculation.

七、部分作业答案

3.1

An error caused by memory allocation can be addressed by allocating adequate memory to the array.

3.2

Shallow copy can lead to “invalid memory access” if the memory referenced is deleted.

3.3

Memory leak caused by not deleting dynamically allocated memory can be addressed by deleting allocated memory after usage.

4.1

Trapezoidal rule is faster.

5.1

By getting rid of duplicated calculations, performance can be improved.