

**课 程 实 验 报 告**

**课程名称： 并行编程原理与实践**

**专业班级： CS1602**

**学 号： U201614545**

**姓 名： 谭胜克**

**指导教师：**

**报告日期： 2019/7/22**

**计算机科学与技术学院**

**目录**

[1串行环境下斐波纳切数列计算 1](#_Toc14700021)

[1.1实验目的与要求 1](#_Toc14700022)

[1.2算法描述 1](#_Toc14700023)

[1.3实验方案 1](#_Toc14700024)

[1.4实验结果与分析 1](#_Toc14700025)

[2 pthread环境斐波纳切数列计算 2](#_Toc14700026)

[2.1实验目的与要求 2](#_Toc14700027)

[2.2算法描述 2](#_Toc14700028)

[2.3 实验方案 2](#_Toc14700029)

[2.4 实验结果与分析 2](#_Toc14700030)

[3 OpenMP环境下斐波纳切数列计算 3](#_Toc14700031)

[3.1实验目的与要求 3](#_Toc14700032)

[3.2算法描述 3](#_Toc14700033)

[3.3实验方案 3](#_Toc14700034)

[3.4实验结果与分析 3](#_Toc14700035)

[4 MPI环境下斐波纳切数列计算 4](#_Toc14700036)

[4.1 实验目的与要求 4](#_Toc14700037)

[4.2 算法描述 4](#_Toc14700038)

[4.3 实验方案 4](#_Toc14700039)

[4.4 实验结果与分析 4](#_Toc14700040)

[5 CUDA环境下斐波纳切数列计算 5](#_Toc14700041)

[5.1 实验目的与要求 5](#_Toc14700042)

[5.2 算法描述 5](#_Toc14700043)

[5.3 实验方案 5](#_Toc14700044)

[5.4 实验结果与分析 5](#_Toc14700045)

[6 实验小结 6](#_Toc14700046)

[**附件：源代码** 7](#_Toc14700047)

# 1串行环境下斐波纳切数列计算

## 1.1实验目的与要求

1. 实验目的
2. 掌握串行环境下斐波那契数列的计算方法。
3. 为后续并行环境下的实验打好基础。
4. 实验要求
5. 在串行环境下编写计算斐波那契数列的C语言小程序，并按要求输出对应的斐波那契数列。
6. 输入: 需要输出的斐波那契额数列的长度，类型为int，如4。
7. 输出: 打印出斐波那契数列，如输入为4，输出为1 1 2 3。
8. 注意: 打印完斐波那契数列之后要换行。

## 1.2算法描述

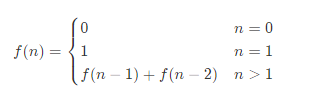


图1.1 斐波那契数列递推公式

根据图1.1所示的递推公式，可以通过递归完成计算，也可以通过非递归的方式进行。我选择了非递归的方式，减少递归过程中递归的开销。通过循环实现也很简单，根据n的值分别按照递推公式进行处理即可。

当n=0时或n=1时，无需计算，直接返回n的值；

当n>1时，通过循环不断将前两项相加的值赋予第三项，之后在把前两项的值往后移一位，循环n-1次即可得到第n项的值。如图1.2所示，为计算部分的核心代码。

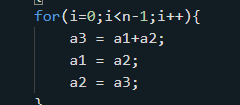


图1.2 串行计算斐波那契数列

## 1.3实验方案

1. 在本机环境下编写代码并测试正确性。
2. 将代码导入平台进行平台测试。
3. 分析代码执行的时间、空间复杂度。

## 1.4实验结果与分析

平台测试结果如图1.3所示。5个测试集均正确完成测试。



图1.3 平台测试结果

# 2 pthread环境斐波纳切数列计算

## 2.1实验目的与要求

1. 实验目的
2. 掌握pthread环境下斐波那契数列的计算方法。
3. 理解pthread并行的实现原理
4. 实验要求
5. 在pthread环境下编写计算斐波那契数列的C语言小程序，并按要求输出对应的斐波那契数列。
6. 输入: 需要输出的斐波那契额数列的长度，类型为int，如4。
7. 输出: 打印出斐波那契数列，如输入为4，输出为1 1 2 3。
8. 注意: 打印完斐波那契数列之后要换行。

## 2.2算法描述

对于斐波那契额数列的计算存在一个通项公式。

pthread环境下新申请一个线程

## 2.3 实验方案

## 2.4 实验结果与分析

# 3 OpenMP环境下斐波纳切数列计算

## 3.1实验目的与要求

1. 实验目的
2. 掌握OpenMP环境下斐波那契数列的计算方法。
3. 理解OpenMP并行的实现原理
4. 实验要求
5. 在OpenMP环境下编写计算斐波那契数列的C语言小程序，并按要求输出对应的斐波那契数列。
6. 输入: 需要输出的斐波那契额数列的长度，类型为int，如4。
7. 输出: 打印出斐波那契数列，如输入为4，输出为1 1 2 3。
8. 注意: 打印完斐波那契数列之后要换行。

## 3.2算法描述

## 3.3实验方案

## 3.4实验结果与分析

# 4 MPI环境下斐波纳切数列计算

## 4.1 实验目的与要求

1. 实验目的
2. 掌握MPI环境下斐波那契数列的计算方法。
3. 理解MPI并行的实现原理
4. 实验要求
5. 在MPI环境下编写计算斐波那契数列的C语言小程序，并按要求输出对应的斐波那契数列。
6. 输入: 需要输出的斐波那契额数列的长度，类型为int，如4。
7. 输出: 打印出斐波那契数列，如输入为4，输出为1 1 2 3。
8. 注意: 打印完斐波那契数列之后要换行。

## 4.2 算法描述

## 4.3 实验方案

## 4.4 实验结果与分析

# 5 CUDA环境下斐波纳切数列计算

## 5.1 实验目的与要求

1. 实验目的
2. 掌握CUDA环境下斐波那契数列的计算方法。
3. 理解CUDA并行的实现原理
4. 实验要求
5. 在CUDA环境下编写计算斐波那契数列的C语言小程序，并按要求输出对应的斐波那契数列。
6. 输入: 需要输出的斐波那契额数列的长度，类型为int，如4。
7. 输出: 打印出斐波那契数列，如输入为4，输出为1 1 2 3。
8. 注意: 打印完斐波那契数列之后要换行。

## 5.2 算法描述

## 5.3 实验方案

## 5.4 实验结果与分析

# 6 实验小结

**附件：源代码**

pthread：

#include <stdio.h>

#include <pthread.h>

#include <math.h>

#include <stdlib.h>

#define MAX\_NUM 100

struct Data

{

int num;

long int fibo[MAX\_NUM];

};

/\*\*\*\*\* Begin \*\*\*\*\*/

void \*fibonacci(void \*data){

struct Data \*tmp = (struct Data\*)data;

int i;

double temp\_1 = (1+sqrt(5))/2;

double temp\_2 = (1-sqrt(5))/2;

double temp\_3 = 1/sqrt(5.0);

tmp->fibo[0] = 1;

tmp->fibo[1] = 1;

for(i=2;i<51;i++){

tmp->fibo[i] = (long int)(temp\_3\*(pow(temp\_1,i+1)-pow(temp\_2, i+1)));

}

}

int main()

{

struct Data data;

pthread\_t th;

int ret;

int num;

int i;

//create thread

ret = pthread\_create(&th, NULL, fibonacci, (void \*)&data);

pthread\_join(th, NULL);

//get input

scanf("%d",&num);

getchar();

//output

for(i = 0; i < num; i++){

if(i == num - 1){

printf("%ld", data.fibo[i]);

}

else{

printf("%ld ", data.fibo[i]);

}

}

printf("\n");

return 0;

}

/\*\*\*\*\* End \*\*\*\*\*/