北京郵電大學

# 实验报告


题目：实验一：基础编程与算法理解实验

班　　级：2022211320

学　　号：2022211683

姓　　名：张晨阳

学　　院：计算机学院（国家示范性软件学院）

2023 年 10 月 6 日

# 一、实验目的

1. 熟悉 C 语言结构的基本操作，掌握对文件的基本操作；
2. 熟悉 C 语言数组结构的存储方式，理解数据的处理方式；
3. 理解算法时间复杂度、执行效率和性能之间的关系；
4. 学习自己查找相关资料以解决实际问题的能力。

# 二、实验环境

Windows 11, Visual Studio Code，x86-64 gcc 10.3

# 三、实验内容

## 实验内容一：结构体及文件基本操作

**要求：**

- 定义一个结构数组，用于保存学生信息，数据项包括：学号、姓名、年龄。
- 功能项1：从键盘输入学生信息存入结构数组，可按照学号依次输入从本人开始的5名同学信息
- 功能项2：输入命令可将结构数组中的数据保存在 input.dat 文件中（在文件中以二进制或文本存储信息均可）
- 功能项3： 输入命令可将信息从 input.dat 文件中读入到结构数组，然后反向顺序输出到 output.dat 文件中（在文件中以二进制或文本存储信息均可）
- 附加要求：设置条件断点，在执行到处理第5位学生信息时中断。

## 实验内容二：算法执行效率测量与分析

**要求：**

- 编写程序分别调用下述两个函数 copyij 和 copyji
- 统计两个子程序的运行时间（绝对时间）
- 比较两个子程序运行绝对时间上的差异，试分析成因
- 给出两个算法的时间复杂度，说明差异
- 附加要求：编译调试时采用优化和非优化方式进行分别进行编译测试，观察启用编译优化后是否能带来性能变化。

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

# 四、实验步骤

# 实验内容一：

1. 运行程序

```
D:\VSCODE\C_C++\lab01\Stru
Please select your operation:
1. Enter student information from the keyboard.
2. Save the data to the input.dat file.
3. Read information from input.dat file into a structure array, reverse the order and output it to output.dat file.
4. Exit.
Enter:
```

2. 输入选项 1，从键盘输入五个学生信息。

```
D:\VSCODE\C_C++\lab01\Stru
Please select your operation:
1. Enter student information from the keyboard.
2. Save the data to the input.dat file.
3. Read information from input.dat file into a structure array, reverse the order and output it to output.dat file.
4. Exit.
Enter:1
Please enter the information of 5 students:
ID of student1: 2023001
NAME of student1: Alice
AGE of student1: 20
ID of student2: 2023002
NAME of student2: Bob
AGE of student2: 21
ID of student3: 2023003
NAME of student3: Charlie
AGE of student3: 22
ID of student4: 2023004
NAME of student4: David
AGE of student4: 23
ID of student5: 2023005
NAME of student5: Emily
AGE of student5: 24
DONE!

Please select your operation:
1. Enter student information from the keyboard.
2. Save the data to the input.dat file.
3. Read information from input.dat file into a structure array, reverse the order and output it to output.dat file.
4. Exit.
Enter:
```

3. 输入选项 2，将结构数组中的数据保存在 input.dat 文件中。

```
Please select your operation:
1. Enter student information from the keyboard.
2. Save the data to the input.dat file.
3. Read information from input.dat file into a structure array, reverse the order and output it to output.dat file.
4. Exit.
Enter:2
DONE!
```

4. 输入选项 3，将信息从 input.dat 文件中读入到结构数组，然后反向顺序输出到 output.dat 文件中。

```
Please select your operation:
1. Enter student information from the keyboard.
2. Save the data to the input.dat file.
3. Read information from input.dat file into a structure array, reverse the order and output it to output.dat file.
4. Exit.
Enter:3
DONE!
```

5. 输入选项 4，结束程序。

6. 检查两个文件是否写入成功。

input.dat | output.dat

文件 编辑 查看

```
2023001   Alice             20 2023002   Bob               21
2023003   Charlie           22 2023004   David             23
2023005   Emily             24
```

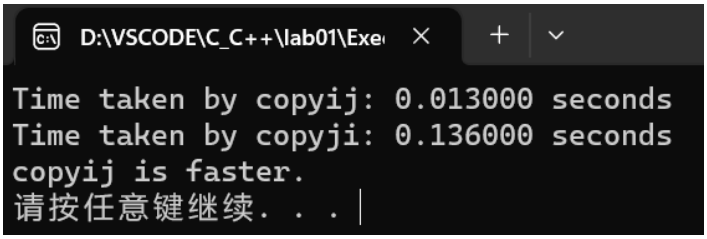
input.dat | output.dat

文件 编辑 查看

```
2023005   Emily             24 2023004   David             23
2023003   Charlie           22 2023002   Bob               21
2023001   Alice             20
```

# 实验内容二：

1. 直接用 vscode 运行程序，发现 copyij 运行更快




2. 运行五次取平均值

| 函数 | 1 | 2 | 3 | 4 | 5 | 平均值 |
|---|---|---|---|---|---|---|
| copyij | 0.013 | 0.016 | 0.013 | 0.012 | 0.010 | 0.013 |
| copyji | 0.136 | 0.122 | 0.133 | 0.120 | 0.116 | 0.125 |

1. 运行时间比较分析

   很明显，`copyij` 的运行比 `copyji` 快。分析可能原因：

   1.1. **内存访问模式：**

   - `copyij` 方法按照行主序复制矩阵，这意味着它在内存中连续访问数组元素，有利于缓存局部性。
   - `copyji` 方法按照列主序复制矩阵，这导致不连续的内存访问，可能导致缓存未命中，从而降低性能。

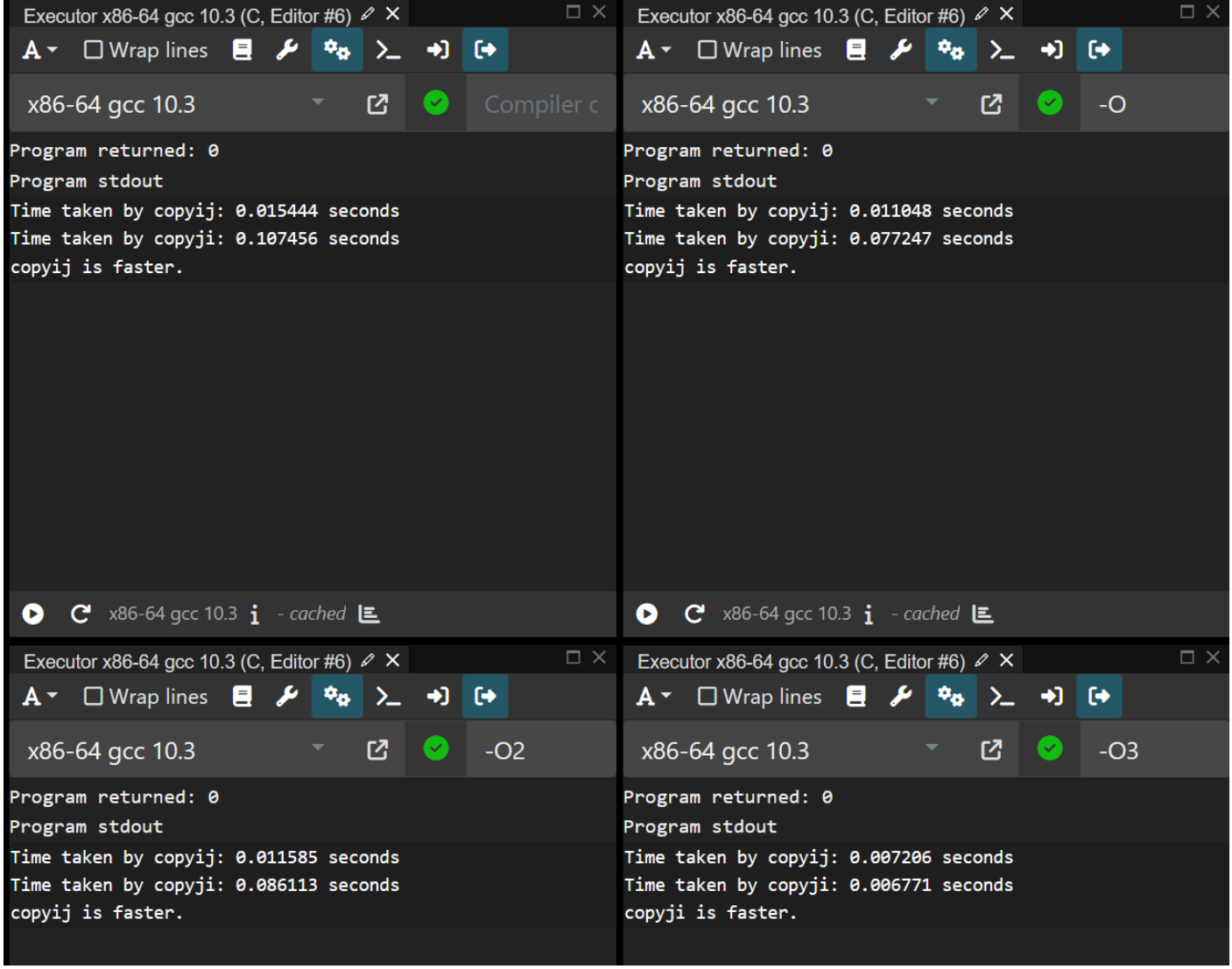
   1.2. **缓存优化：**

   - 现代计算机通常具有多级缓存（例如 L1、L2和 L3缓存），这些缓存用于存储最近使用的数据。`copyij` 的行主序访问方式更有可能利用这些缓存，因为它倾向于访问连续的内存位置，减少了缓存未命中的可能性。

2. 时间复杂度分析

   二者时间复杂度都是 $O(n^2)$，所以时间复杂度不是导致运行时间不同的主要原因。

3. 优化与否带来的性能变化分析




当不采用优化或采用 O1，O2 优化时，`copyij` 运行比 `copyji` 快，但采用 O3 优化时，`copyji` 比 `copyij` 快，但二者较为接近。可能原因分析：

#### 3.1. 循环展开（Loop Unrolling）：

- 编译器在O3优化级别下可能会执行循环展开，这意味着它会将循环体内的代码复制多次，以减少循环迭代的开销。在 `copyji` 中，列主序复制可能更容易进行循环展开，因为每次复制操作涉及到连续的内存访问，而在 `copyij` 中，行主序复制可能更难以进行循环展开。

#### 3.1. 寄存器分配：

- O3级别的优化通常包括更好的寄存器分配策略。寄存器分配可以使得程序更有效地使用CPU寄存器，减少内存访问的次数。在某些情况下，`copyji` 的列主序复制可能更容易进行寄存器分配。

# 五、实验分析和总结

## 实验内容一：

1. **工作思路：**

首先需要定义一个结构体；对于三个功能项，分别定义三个函数执行；加上第四个功能项，退出程序；通过 vscode 直接设置条件断点。

#### 1.1. 功能项1 - 从键盘输入学生信息：

- 使用循环来依次输入5名学生的信息，可以通过 `for` 循环来实现。
- 在每次循环中，使用 `scanf` 或其他输入函数来获取学号、姓名和年龄，并将其存储到结构数组中的相应位置。

#### 1.1. 功能项2 - 将信息保存到文件：

- 创建一个文件指针，用于打开一个文件以写入学生信息。
- 使用 `fwrite` 等函数将结构数组中的数据写入文件中。

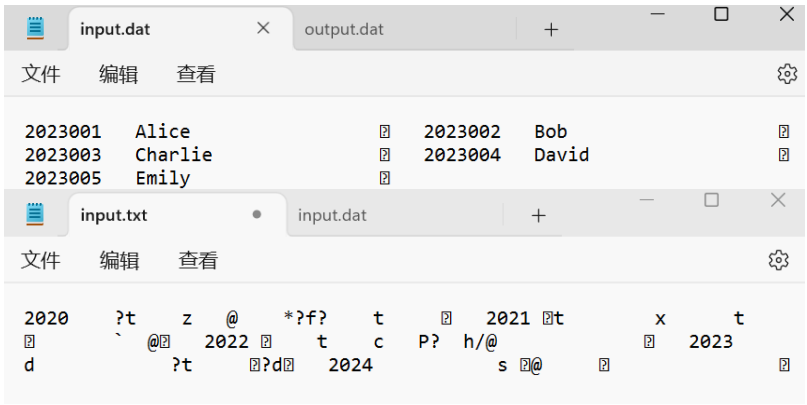
#### 1.2. 功能项3 - 从文件中读取信息并反向输出到文件：

- 创建一个文件指针，用于打开 `input.dat` 文件以读取学生信息。
- 使用 `fread` 等函数将数据从文件中读入到结构数组中。
- 在读取完成后，可以使用 `for` 循环反向输出学生信息到另一个文件中。

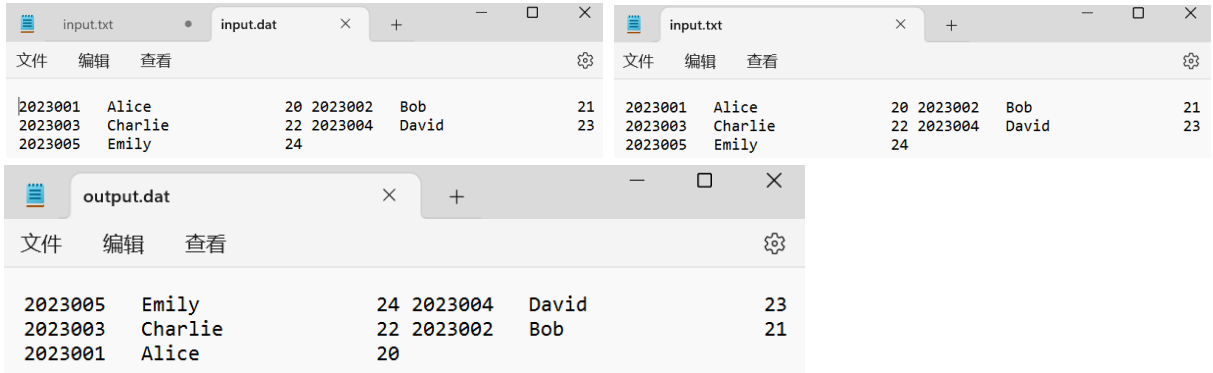
2. **问题分析：**

在使用 `fwrite` 函数时，如果输入类型为 int，则会出现乱码，原因是写入文件是二进制代码形式；当学号、姓名、年龄都用字符串表示，则不会出现乱码。

fwrite 效果（输入中有 int）：




fwrite 效果（都用字符串输入）：




# 实验内容二：

1. 工作思路

1.1. 首先，编写一个包含这两个函数的程序，并在主函数中调用它们。同时，使用标准的计时函数 `clock()` 在程序中测量每个函数的运行时间。再通过 `elapsed_time_ = ((double)(end_time - start_time)) / CLOCKS_PER_SEC;` 将两个时间点之间的时间间隔以秒为单位存储

1.2. 运行性能测试：运行程序并记录两个函数的运行时间。确保多次运行以获得稳定的平均值。

1.3. 分析性能差异：比较两个函数的运行时间并分析性能差异的原因。在这两个函数中，`copyij` 使用了嵌套的循环来复制数组，而 `copyji` 则使用了不同的循环顺序。这两种循环顺序可能会导致不同的内存访问模式，影响缓存效率。考虑以下因素：

- 缓存命中率：一种循环顺序可能导致更高的缓存命中率，从而更快的执行时间。
- 内存访问模式：不同的循环顺序可能导致不同的内存访问模式，从而影响内存带宽的利用率。

1.4. 时间复杂度分析：分析两个算法的时间复杂度。在 `copyij` 中，两个嵌套循环的时间复杂度为 $O(n^2)$。在 `copyji` 中，循环的次序导致时间复杂度仍然为 $O(n^2)$。

1.5. 测试编译选项：分别使用优化和非优化的编译选项来编译程序，并比较使用优化和非优化编译选项的性能结果。

2. **心得总结：**

为了避免数组读取效率带来的影响，对于两个 copy 函数，分别定义两组数组进行实验，而不是在同一个数组上重复操作。

# 六、程序源代码

## 实验内容一：

```c
#include <stdio.h>
#include <stdlib.h>

struct Student
{
    char studentID[10];
    char name[20];
    // int age; // 会出现乱码
    char age[3];
} students[5];

// 功能项1：从键盘输入学生信息
void input_five()
{
    printf("Please enter the information of 5 students:\n");
    for (int i = 0; i < 5; i++)
    {
        printf("ID of student%d: ", i + 1);
        scanf("%s", students[i].studentID);
        printf("NAME of student%d: ", i + 1);
        scanf("%s", students[i].name);
        printf("AGE of student%d: ", i + 1);
        scanf("%d", &students[i].age);
    }
    printf("DONE!\n\n");
}

// 功能项2：将学生信息保存到input.dat文件中
void data_sav()
{
    FILE *file;
    file = fopen("input.dat", "wb");
    if (file != NULL)
    {
        fwrite(students, sizeof(struct Student), 5, file);
        fclose(file);
        printf("DONE!\n\n");
    }
    else
    {
        printf("cannot open file input.dat\n");
        exit(EXIT_FAILURE);
    }
}

// 功能项3：从input.dat文件中读取学生信息并反向顺序输出到output.dat文件中
```

```c
void reverse()
{
    FILE *fpin, *fpout;
    fpin = fopen("input.dat", "rb");
    fpout = fopen("output.dat", "wb");
    if (fpin != NULL && fpout != NULL)
    {
        fseek(fpin, 0L, SEEK_END); // 将文件指针移动到文件末尾
        long size = ftell(fpin);   // 获取文件大小
        rewind(fpin);              // 将文件指针重置到文件开头

        int numStudents = size / sizeof(struct Student); // 计算学生信息的数量
        struct Student *students = (struct Student *)malloc(sizeof(struct
Student) * numStudents);
        if (students == NULL)
        {
            printf("内存分配失败\n");
            return;
        }

        // 读取学生信息到内存中
        fread(students, sizeof(struct Student), numStudents, fpin);
        for (int i = numStudents - 1; i >= 0; i--)
        {
            // 反向顺序写入到output.dat文件中
            fwrite(&students[i], sizeof(struct Student), 1, fpout);
        }

        free(students);
        fclose(fpin);
        fclose(fpout);
        printf("DONE!\n\n");
    }
    else
    {
        printf("cannot open file input.dat or output.dat\n");
        exit(EXIT_FAILURE);
    }
}

int main()
{
    int choice;
    while (1)
    {
        printf("Please select your operation:\n");
        printf("1. Enter student information from the keyboard.\n");
        printf("2. Save the data to the input.dat file.\n");
        printf("3. Read information from input.dat file into a structure array,
```

```
reverse the order and output it to output.dat file.\n");
        printf("4. Exit.\n");
        printf("Enter:");
        scanf("%d", &choice);

        if (choice == 1)
            input_five();
        else if (choice == 2)
            data_sav();
        else if (choice == 3)
            reverse();
        else
            break;
    }
    return 0;
}
```

## 实验内容二：

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 2048

int src1[N][N];
int src2[N][N];
int dst1[N][N];
int dst2[N][N];

void copyij(int src[2048][2048], int dst[2048][2048])
{
    int i, j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}

void copyji(int src[2048][2048], int dst[2048][2048])
{
    int i, j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}

int main()
{
    clock_t start_time, end_time; // 存储时间值
    double elapsed_time_ij, elapsed_time_ji;

    for (int i = 0; i < 2048; i++)
        for (int j = 0; j < 2048; j++)
            src2[i][j] = src1[i][j] = i + j;
    // 测试copyij 函数
    start_time = clock();
    copyij(src1, dst1);
    end_time = clock();
    elapsed_time_ij = ((double)(end_time - start_time)) / CLOCKS_PER_SEC;

    // 测试 copyji 函数
    start_time = clock();
    copyji(src2, dst2);
    end_time = clock();
    elapsed_time_ji = ((double)(end_time - start_time)) / CLOCKS_PER_SEC;
```

```
    // 输出运行时间
    printf("Time taken by copyij: %f seconds\n", elapsed_time_ij);
    printf("Time taken by copyji: %f seconds\n", elapsed_time_ji);

    // 比较运行时间
    if (elapsed_time_ij < elapsed_time_ji)
        printf("copyij is faster.\n");
    else
        printf("copyji is faster.\n");

    system("pause");
    return 0;
}
```