12.07 实验课

2023.12.07 程设实验课

普通作业 - 指针类型转换

```
float x = 0.3;
cout << *((int *)&x) << endl;
```

普通作业 - 深入理解 Struct 内存布局

Notice：这个讲解和本周普通作业的链表结构分析强强相关，请大家认真听（会有点难🐶）

```
#include <iostream>

struct {
    unsigned short b1 : 10;
    unsigned short b2 : 6;
    unsigned short b3 : 16;
} S;

int main() {
    std::cout << sizeof(S) << '\n';
}
```

```
#include <iostream>

struct {
    unsigned short b1 : 10;
    unsigned short b2 : 16;
    unsigned short b3 : 6;
} S;

int main() {
    std::cout << sizeof(S) << '\n';
}
```

问题 0: 这个结构体啥意思?

冒号后的数字表示这个结构体占的 bit 数

问题 1: 如何查看一段内存空间中每个byte的值?

把这段内存空间映射到 byte

什么数据类型可以表示一个byte? -> char

```
unsigned char* p = (unsigned char*)malloc(sizeof(S));

unsigned char* t = p;
while (t < p + sizeof(S)) {
    printf("%02x ", *t);
    t++;
}
cout << endl;
```

问题 2: 如何查看每个字段都占的啥玩意

把这段内存空间映射成 S 结构

```
S* s = (S*) p;
```

怎么安全的转换? 万一我这段代码的内存空间分配并不足以给S呢? C++ 提供了一个安全的类型转换方法 static_cast，请你试一下下面的代码

```
S* ts = static_cast<S*>(p);
```

留一个思考题，我们如何让 `static_cast` 方法相信我们可以正确的转换?

问题 3: 如何标记这段内存空间被用了?

把它用的所有内存都标记为 1，既然全都是 0，那就全部取反

```
s->b1 = (~s->b1);
```

```
s->b2 = (~s->b2);
s->b3 = (~s->b3);
```

问题 4: 怎么检查这些变量都是个啥?

用二进制的形式去检查

```
printf("b1: %02x\n", s->b1);
printf("b2: %02x\n", s->b2);
printf("b3: %02x\n", s->b3);
```

问题 5: 怎么查看结构体内部空间整体是个啥?

也是用二进制的形式去检查

```
unsigned char* t = p;
while (t < p + sizeof(S)) {
    printf("%02x ", *t);
    t++;
}
cout << endl;
```

问题 6: 这样两个结构体内部是什么个情况?

```
// ff ff ff ff
struct S {
    unsigned short b1 : 6;
    unsigned short b2 : 10;
    unsigned short b3 : 16;
};

// ff 03 ff ff 3f 00
struct S{
    unsigned short b1 : 10;
    unsigned short b2 : 16;
    unsigned short b3 : 6;
}
```

问题 7: 这个输出的bit结果是不是不太对劲?

可以看到，在内存对齐后的 6byte 结构体里，我们看到的应该是 `03 ff ff ff 00 3f`

但为什么我们拿到的是 `ff 03 ff ff 3f 00`

- （听个乐）《计算机组成原理》《CSAPP》 malloc 分配的内存地址在堆里，堆内数据是从高地址向低地址的，和栈相反

```
t = p;
while (t < p + sizeof(S)) {
    printf("%llx: ", (unsigned long long)(t));
    printf("%02x\n", *t);
    t++;
}
```

```
326060a0: ff
326060a1: 03
326060a2: ff
326060a3: ff
326060a4: 3f
326060a5: 00
```

问题 8: 内存对齐的规则是什么?

1. 第一个成员的首地址为 0
2. 每个成员的首地址是自身大小的整数倍
3. 结构体的总大小，为其成员中所含最大类型的整数倍

下面这个结构体的内存占用大小是多少字节?

```
struct Node{
    int data;
    Node* next;
}
```

问题 9: 我们可以强制它紧密排布吗?

如果是量化交易，按照微秒算钱的场景，每个 bit 都价值千金

```
#pragma pack(1)
```

```
struct S {
    unsigned short b1 : 10;
    unsigned short b2 : 16;
    unsigned short b3 : 6;
};
```

使用 `pragma pack(1)` 让 C++ 按照 1 bit 进行内存对齐

整体分析代码

```
#include <iostream>
using namespace std;

#pragma pack(1)
struct S {
    unsigned short b1 : 10;  // 0x3ff = 1111111111
    unsigned short b2 : 16;  // 0xffff = 1111111111111111
    unsigned short b3 : 6;   // 0x3f = 111111
    // ff 03 ff ff 3f 00
    // 03 ff ff ff 00 3f
};

int main() {
    cout << "size of S: " << sizeof(S) << endl;
    unsigned char* p = (unsigned char*)malloc(sizeof(S));
    S* ts = static_cast<S*>(static_cast<void*>(p));

    unsigned char* t = p;
    while (t < p + sizeof(S)) {
        printf("%02x ", *t);
        t++;
    }
    cout << endl;

    S* s = (S*)p;
    printf("b1: %02x\n", s->b1);
    printf("b2: %02x\n", s->b2);
    printf("b3: %02x\n", s->b3);

    s->b1 = (~s->b1);
    s->b2 = (~s->b2);
    s->b3 = (~s->b3);
    cout << endl;

    printf("b1: %02x\n", s->b1);  // 0x3ff = 1111111111
```

```
    printf("b2: %02x\n", s->b2);  // 0xffff = 1111111111111111
    printf("b3: %02x\n", s->b3);  // 0x3f = 111111

    t = p;
    while (t < p + sizeof(S)) {
        printf("%llx: ", (unsigned long long)(t));
        printf("%02x\n", *t);
        t++;
    }
    cout << endl;
}
```

普通作业 - void* 指针

如果指针类型为 `void*`，则该指针可以指向任何未使用 `const` 或 `volatile` 关键字声明的变量。 `void*` 指针不能取消引用，除非它被强制转换为另一种类型。 `void*` 指针可以转换为任何其他类型的数据指针。 —— Microsoft C++ reference

- 该指针可以指向任何未使用 `const` 或 `volatile` 关键字声明的变量
- `S* ts = static_cast<S*>(p);` 不能编译
- `S* ts = static_cast<S*>(static_cast<void*>(p))`

结构化编程基本原则与实践

借书

任务描述

这个题目是关于设计一个图书管理系统，用于追踪学生的借书记录。每个学生和每本书都有相应的信息。初始状态下，所有学生没有借过任何书籍。系统会根据输入的借书记录来跟踪每位学生借阅的书籍情况，并最输出所有学生的借阅信息和相对应的书本。

说明

- 初始状态下，所有学生没有借过任何书籍。
- 每个学生最多借阅三本书。如果一次输入超过三本书，后面的书籍记录将被视为非法，无法借阅。
- 输入格式：
  - 首先输入 N(N<10)，代表N个学生。
  - 下面输入 N 行，每行有学生的姓名name(name长度 < 20)和学号number(number长度 <10)。

- 其次输入 M(M<50)，代表动作记录。
- 包含多个动作记录。每行包含学生的姓名、书名以及借书发生的时间。动作时间以 HH:MM:SS 格式给出。

- 输出格式：
  - 按照学生字符串非递减的顺序输出，对于每位学生，输出其姓名、学号以及所有他借到书籍的信息（如果一个学生借了多本书，输出的记录要按照书名的字符串非递减进行排序，当书名相同时，按照时间进行非递减排序)。

注意事项：

1. 图书馆里有足够的书籍，且每种书籍的数量无限。
2. 字符串比较遵循逐字符比较的方式，首先比较第一个字符，不同则较小编码值的字符排在前面；相同则继续比较下一个字符。短字符串被视为较小。
3. 为了简化题目，所有操作都是在同一天发生的。

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;
const int MAX_BOOKS = 3;
const int MAX_STUDENTS = 10;
const int MAX_ACTIONS = 50;
const int MAX_NAME_LENGTH = 20;
const int MAX_ID_LENGTH = 5;
const int MAX_TITLE_LENGTH = 20;
const int MAX_TIME_LENGTH = 10;

struct Book {
    char title[MAX_TITLE_LENGTH];
    char time[MAX_TIME_LENGTH];
    bool borrowed;
};

struct Student {
    char name[MAX_NAME_LENGTH];
    char id[MAX_ID_LENGTH];
    Book books[MAX_BOOKS];
    int numBooks;
};

bool compareBooks(const Book &a, const Book &b) {
    int cmp = std::strcmp(a.title, b.title);
    if (cmp != 0) {
```

```
        return cmp < 0;
    }
    return std::strcmp(a.time, b.time) < 0;
}

bool compareStudents(const Student &a, const Student &b) {
    return std::strcmp(a.name, b.name) < 0;
}

int main() {
    int N, M;
    std::cin >> N;
    Student students[MAX_STUDENTS];

    for (int i = 0; i < N; ++i) {
        std::cin >> students[i].name >> students[i].id;
        students[i].numBooks = 0;
    }

    std::cin >> M;
    for (int i = 0; i < M; ++i) {
        char studentName[MAX_NAME_LENGTH], bookTitle[MAX_TITLE_LENGTH], time[MAX
        std::cin >> studentName >> bookTitle >> time;

        for (int j = 0; j < N; ++j) {
            if (std::strcmp(students[j].name, studentName) == 0 && students[j].n
                Book newBook;
                std::strcpy(newBook.title, bookTitle);
                std::strcpy(newBook.time, time);
                newBook.borrowed = true;

                students[j].books[students[j].numBooks++] = newBook;
                std::sort(students[j].books, students[j].books + students[j].num
                break;
            }
        }
    }

    std::sort(students, students + N, compareStudents);

    for (int i = 0; i < N; ++i) {
        std::cout << students[i].name << " " << students[i].id << " ";
        for (int j = 0; j < students[i].numBooks; ++j) {
            std::cout << students[i].books[j].title << " " << students[i].books[
            if(j!=students[i].numBooks-1){
                cout << " ";
            }
```

```
        }
        std::cout << std::endl;
    }

    return 0;
}

```

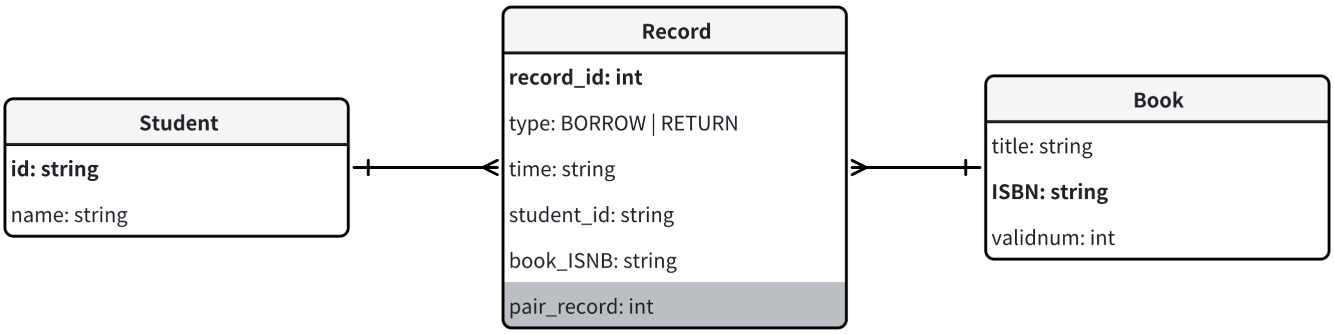
借书 Pro Max

有一个借书系统，每个学生和每本书都有相应的信息，请你设计一个图书管理系统，帮助老师追踪每个同学的借阅信息、每本图书的出借情况和借阅记录、和图书馆整体的馆藏信息

testcase：

```
// 首先录入学生的身份信息
// 姓名 学号
2
A 00
B 11

// 录入书籍的信息
// 书名 ISBN 馆藏数
3
CodeC++ ISBN00 1
CodePython ISBN11 2
CodeC# ISBN22 1

// 录入同学的借阅归还信息
// 学生学号 书籍ISBN 操作时间 YYmmddThhmmss 操作行为
4
00 ISBN00 20231206T080000 borrow
00 ISBN11 20231206T090000 return
00 ISBN00 20231206T090000 return
11 ISBN00 20231206T083000 borrow


// 下面是输出信息
// =============
// 同学的借书信息
A 00
B 11
CodePython ISBN11 20231206T083000

// 书籍的借阅记录
CodeC++ ISBN00
```

```
00 20231206T080000 A
11 20231206T083000 B
CodePython ISBN11
CodeC# ISBN22
```




```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;
const int MAX_BOOKS = 3;
const int MAX_STUDENTS = 10;
const int MAX_RECORDS = 100;

const int MAX_ACTIONS = 50;
const int MAX_NAME_LENGTH = 20;
const int MAX_ID_LENGTH = 5;
const int MAX_TITLE_LENGTH = 20;
const int MAX_TIME_LENGTH = 30;
const int MAX_ISBN_LENGTH = 10;
const int TYPE_BORROW = 1;
const int TYPE_RETURN = 0;

using namespace std;

struct Book
{
    char title[MAX_TITLE_LENGTH];
    char ISBN[MAX_ISBN_LENGTH];
    int validnum;
};

struct Student
{
    char name[MAX_NAME_LENGTH];
    char id[MAX_ID_LENGTH];
```

```
31 };
32
33 struct Record
34 {
35     int record_id;
36     int type;
37     char time[MAX_TIME_LENGTH];
38     char student_id[MAX_ID_LENGTH];
39     char book_ISNB[MAX_ISBN_LENGTH];
40     int pair_record;
41 };
42
43 int next_record_id = 0;
44 int book_num = 0;
45 int student_num = 0;
46 int select_borrow_record(Record records[], char student_id[], char book_ISBN[])
47 {
48     for (int i = 0; i < next_record_id; i++)
49     {
50         if (records[i].type == TYPE_BORROW, strcmp(records[i].student_id, studen
51         {
52             return i;
53         }
54     }
55     return -1;
56 }
57
58 Book *select_book(Book books[], char book_ISBN[])
59 {
60     for (int i = 0; i < book_num; i++)
61     {
62         if (strcmp(books[i].ISBN, book_ISBN) == 0)
63         {
64             return &books[i];
65         }
66     }
67     return nullptr;
68 }
69
70 Student *select_student(Student students[], char student_id[])
71 {
72     for (int i = 0; i < student_num; i++)
73     {
74         if (strcmp(students[i].id, student_id) == 0)
75         {
76             return &students[i];
77         }
```

```
78      }
79      return nullptr;
80  }
81
82  bool add_record(Record records[], Book books[], char student_id[], char book_ISB
83  {
84      if (next_record_id == MAX_RECORDS)
85      {
86          return false;
87      }
88
89      records[next_record_id].record_id = next_record_id;
90      strcpy(records[next_record_id].book_ISNB, book_ISBN);
91      strcpy(records[next_record_id].student_id, student_id);
92      strcpy(records[next_record_id].time, time);
93      records[next_record_id].type = type;
94
95      Book *b = select_book(books, records[next_record_id].book_ISNB);
96      if (type == TYPE_RETURN)
97      {
98          int pair_id = select_borrow_record(records, student_id, book_ISBN);
99          if (pair_id != -1)
100         {
101             records[pair_id].pair_record = next_record_id;
102         }
103         b->validnum += 1;
104     }
105     else
106     {
107         if (b == nullptr)
108         {
109             return false;
110         }
111         if (b->validnum == 0)
112         {
113             return false;
114         }
115         b->validnum -= 1;
116         records[next_record_id].pair_record = -1;
117     }
118     next_record_id++;
119     return true;
120 }
121
122 void output_student_record_info(Student students[], Record records[], Book books
123 {
124     for (int i = 0; i < student_num; ++i)
```

```
125     {
126         cout << students[i].name << " " << students[i].id << endl;
127         for (int j = 0; j < next_record_id; j++)
128         {
129             // cout << j << " " << records[j].student_id << " " << students[j].i
130             if (strcmp(records[j].student_id, students[i].id) == 0)
131             {
132                 if (records[j].type == TYPE_BORROW && records[j].pair_record ==
133                 {
134                     Book *b = select_book(books, records[j].book_ISNB);
135                     if (b != nullptr)
136                     {
137                         cout << b->title << " " << b->ISBN << " " << records[j].
138                     }
139                 }
140             }
141         }
142     }
143 }
144
145 void output_book_record_info(Student students[], Record records[], Book books[])
146 {
147     for (int i = 0; i < book_num; ++i)
148     {
149         cout << books[i].title << " " << books[i].ISBN << endl;
150         for (int j = 0; j < next_record_id; j++)
151         {
152             if (strcmp(records[j].book_ISNB, books[i].ISBN) == 0)
153             {
154                 if (records[j].type == TYPE_BORROW)
155                 {
156                     Student *s = select_student(students, records[j].student_id)
157                     if (s != nullptr)
158                     {
159                         cout << s->id << " " << s->name << endl;
160                     }
161                 }
162             }
163         }
164     }
165 }
166
167 int main()
168 {
169     int N, M, H;
170     cin >> N;
171     student_num = N;
```

```
    Student students[MAX_STUDENTS];
    Book books[MAX_BOOKS];
    Record records[MAX_RECORDS];

    for (int i = 0; i < N; ++i)
    {
        cin >> students[i].name >> students[i].id;
    }


    cin >> M;
    book_num = M;
    for (int i = 0; i < M; ++i)
    {
        cin >> books[i].title >> books[i].ISBN >> books[i].validnum;
    }


    cin >> H;
    char student_id[MAX_ID_LENGTH];
    char book_ISBN[MAX_ISBN_LENGTH];
    char time[MAX_TIME_LENGTH];
    char type_str[20];
    for (int i = 0; i < H; ++i)
    {
        cin >> student_id >> book_ISBN >> time >> type_str;
        if (strcmp(type_str, "borrow") == 0)
        {
            add_record(records, books, student_id, book_ISBN, time, TYPE_BORROW)

        }
        else
        {
            add_record(records, books, student_id, book_ISBN, time, TYPE_RETURN)
        }
    }


    output_student_record_info(students, records, books);
    output_book_record_info(students, records, books);

    return 0;
}

```