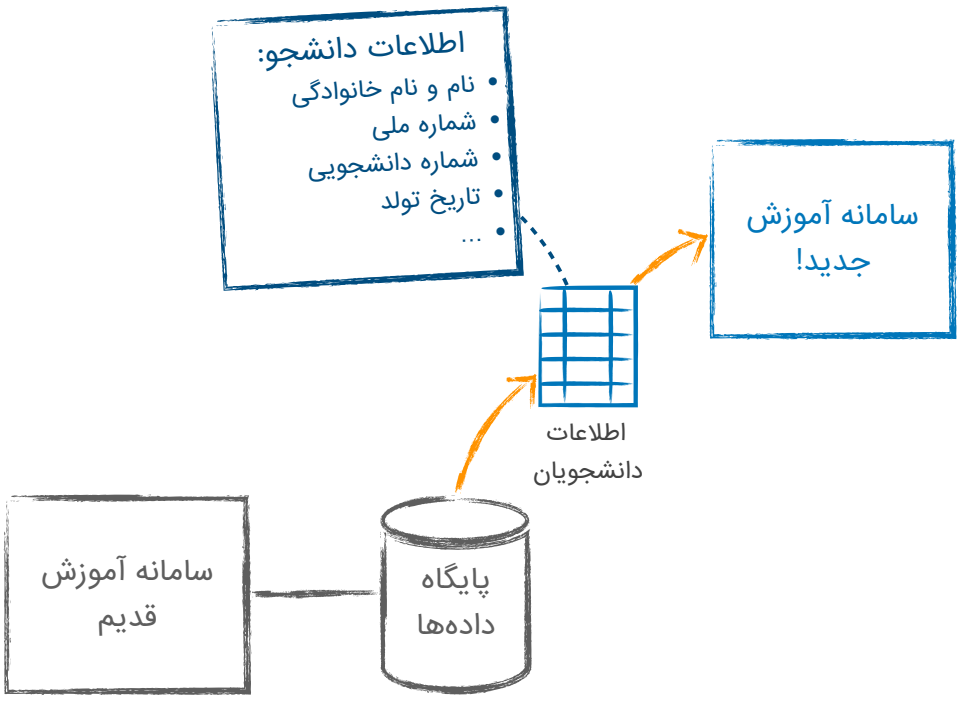
دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

رسیدگی به خطاها — ۳

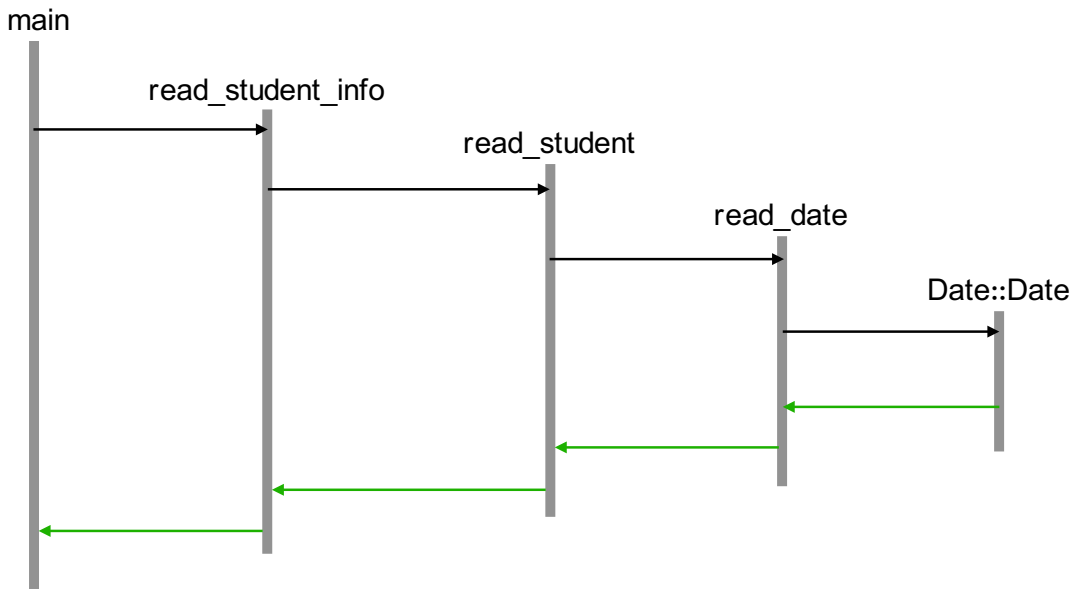
ناکافی‌بودن روش‌های موجود

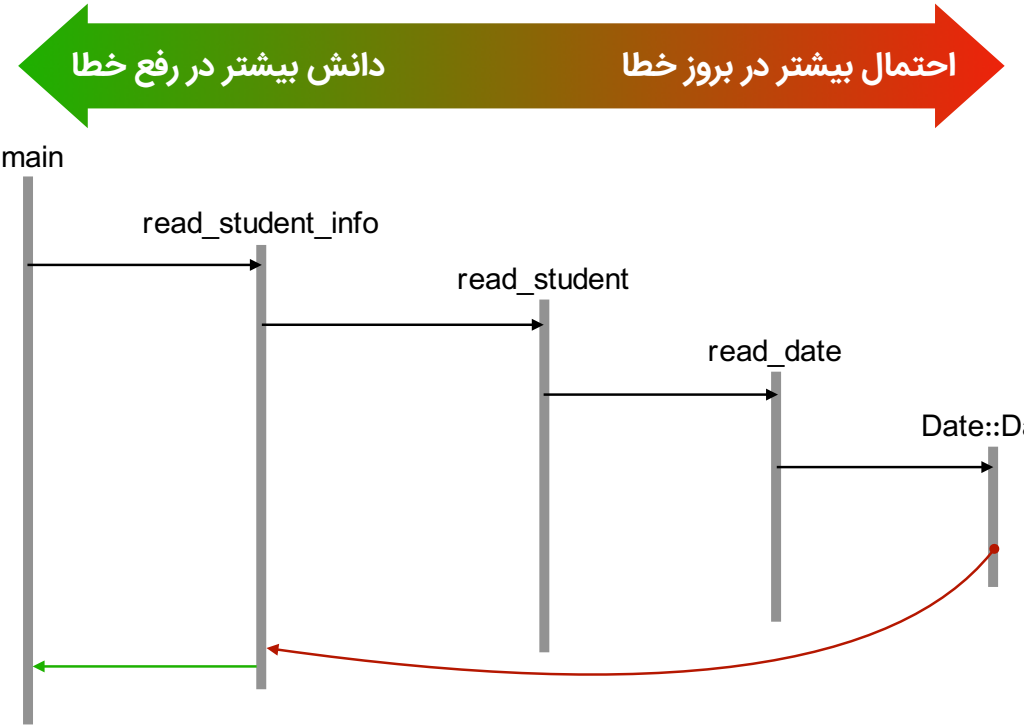
برنامه‌سازی پیشرفته — رامتین خسروی

بهار ۹۹



فراخوانی‌های تودرتوی توابع — اجرای بدون خطا





نیاز به سازوکاری برای انتقال حالت خطا از محل بروز خطا به محل رسیدگی به خطا

استفاده از مقدار بازگشتی تابع

```
Employee* find_employee_by_id(string empid) {
    for (int i = 0; i < employees.size(); i++)
        if (employees[i].get_id() == empid)
            return &employees[i];
    return NULL;
}
```

برای تمام توابع نمی‌توان مقدار بازگشتی «خطا» تعریف کرد.

```
Date read_date(ifstream& input) {
    int d, m, y;
    char ch;        // for reading slash separators ('/')
    input >> d >> ch >> m >> ch >> y;
    return Date(d, m, y);
}
```

چه مقدار خاصی از تایپ تاریخ می‌تواند نشان‌دهنده خطا باشد؟

https://github.com/ramtung/apnotes/tree/master/10_Exceptions/Import.2-ErrorHandlingByRef

```cpp
Date::Date(int d, int m, int y)
{
    if (y < 0 || m < 1 || m > 12 || d < 1 || d > days_of_month(m, y))
        abort();

    day = d;
    month = m;
    year = y;
}
```

اساساً سازنده مقداری برنمی‌گرداند

راه دوم: استفاده از پارامتر رد شده با ارجاع

```cpp
Date read_date(ifstream& input, int& result);
```

راه دوم: استفاده از پارامتر رد شده با ارجاع

```
Date read_date(ifstream& input, int& result) {
    int d, m, y;
    char ch;
    input >> d;
    input >> ch;
    if (ch != '/') {
        // ???
    }
    input >> m;
    input >> ch;
    if (ch != '/') {
        // ???
    }
    input >> y;
    return Date(d, m, y);
}
```

date.h

```
const int DATE_OK = 0;
const int DATE_INVALID_SEPARATOR = -1;
```

date.cpp

```
Date read_date(ifstream& input, int& result) {
    int d, m, y;
    char ch;
    input >> d;
    input >> ch;
    if (ch != '/') {
        result = DATE_INVALID_SEPARATOR;
        return ???;
    }
    input >> m;
    input >> ch;
    if (ch != '/') {
        result = DATE_INVALID_SEPARATOR;
        return ???;
    }
    input >> y;
    result = DATE_OK;
    return Date(d, m, y);
}
```

date.h

```
const int DATE_OK = 0;
const int DATE_INVALID_SEPARATOR = -1;
```

date.cpp

```
Date read_date(ifstream& input, int& result) {
    int d, m, y;
    char ch;
    input >> d;
    input >> ch;
    if (ch != '/') {
        result = DATE_INVALID_SEPARATOR;
        return Date(1,1,1);
    }
    input >> m;
    input >> ch;
    if (ch != '/') {
        result = DATE_INVALID_SEPARATOR;
        return Date(1,1,1);
    }
    input >> y;
    result = DATE_OK;
    return Date(d, m, y);
}
```

بروز خطا

```
Student read_student(ifstream& input, int& result) {
    string name;
    input >> name;
    int date_result;
    Date bdate = read_date(input, date_result);
    if (date_result != DATE_OK) {
        result = STUDENT_ERROR;
        return Student("", bdate);
    }
    result = STUDENT_OK;
    return Student(name, bdate);
}
```

انتشار خطا

چه تضمینی وجود دارد که برنامه‌نویس خطا را کنترل و منتشر کند؟

```
void read_student_info(char* filename, vector<Student>& v) {
    ifstream input(filename);
    int count;
    input >> count;
    for (int i = 0; i < count; i++) {
        int std_result;
        Student s = read_student(input, std_result);
        if (std_result == STUDENT_OK)
            v.push_back(s);
        else {
            input.clear();
            string to_be_ignored;
            getline(input, to_be_ignored);
        }
    }
    input.close();
}
```

رسیدگی به خطا

رسیدگی به خطا در سازنده

```
Date::Date(int d, int m, int y, int& result)
{
    if (y < 0 || m < 1 || m > 12 || d < 1 || d > days_of_month(m, y))
    {
        result = DATE_INVALID_ARGS;
        day = month = year = 1;
        return;
    }
    result = DATE_OK;
    day = d;
    month = m;
    year = y;
}
```

و باز هم ناخواناتر شدن کد

```
Date read_date(ifstream& input, int& result) {
    int d, m, y;
    char ch;
    int constructor_result;
    result = DATE_OK;
    input >> d;
    input >> ch;
    if (ch != '/') {
        result = DATE_INVALID_SEPARATOR;
        return Date(1,1,1, constructor_result);
    }
    input >> m;
    input >> ch;
    if (ch != '/') {
        result = DATE_INVALID_SEPARATOR;
        return Date(1,1,1, constructor_result);
    }
    input >> y;
    return Date(d, m, y, result);
}
```

مشکلات

- اضافه‌شدن یک پارامتر به تمام توابع و سازنده‌ها
- نیاز به تعریف متغیرهای result
- اضافه‌شدن یک if به ازای هر فراخوانی که احتمال خطا دارد
- مقادیر بازگشت نامعقول

⟸ ناخوانا شدن بی‌دلیل متن برنامه

راه سوم: استفاده از متغیر سراسری خطا

```
Date read_date(ifstream& input) {
    date_result = DATE_OK;
    int d, m, y;
    char ch;
    input >> d;
    input >> ch;
    if (ch != '/') {
        date_result = DATE_INVALID_SEPARATOR;
        return Date(1,1,1);
    }
    input >> m;
    input >> ch;
    if (ch != '/') {
        date_result = DATE_INVALID_SEPARATOR;
        return Date(1,1,1);
    }
    input >> y;
    return Date(d, m, y);
}
```

مشکلات

- ~~اضافه‌شدن یک پارامتر به تمام توابع و سازنده‌ها~~
- ~~نیاز به تعریف متغیرهای result~~
- اضافه‌شدن یک if به ازای هر فراخوانی که احتمال خطا دارد
- مقادیر بازگشت نامعقول

⟸ ناخوانا شدن بی‌دلیل متن برنامه

به سازوکاری برای رسیدگی به خطا نیاز داریم که:

- خطا را به طور خودکار تا نقطه رسیدگی منتشر کند
- ما را مجبور به اضافه‌کردن کدهای بی‌مورد نکند

تمرین‌های کوتاه

۱. در تابع `read_date` به خطاهای مربوط به خواندن روز و ماه و سال از ورودی نیز رسیدگی نمایید.

راهنمایی: بروز خطا در خواندن از یک جریان ورودی مثل `input` را می‌توان با `if(!input)` چک کرد.

۲. در سازنده `Student` نام وارد شده نباید تهی باشد. به این خطا نوعی رسیدگی کنید که در صورت مواجهه با چنین سطری خواندن فایل ورودی متوقف شود اما سطرهایی که تا قبل از آن خوانده شده پردازش شوند.