دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

رسیدگی به خطاها — ۶

بازپرتاب استثناءها

برنامه‌سازی پیشرفته — رامتین خسروی

بهار ۹۹

read_student_info

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

read_student

```
Student read_student(ifstream& input)
{
  string name;
  input >> name;
  Date bdate = read_date(input);
  return Student(name, bdate);
}
```

read_date

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

read_student_info

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

read_student

```
Student read_student(ifstream& input)
{
  string name;
  input >> name;
  Date bdate = read_date(input);
  return Student(name, bdate);
}
```

read_date

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

read_student_info

```
void read_student_info(char* filename,
        vector<Student>& v)
{
  ifstream input(filename);
  int count;
  input >> count;
  for (int i = 0; i < count; i++) {
    try {
      Student s = read
      v.push_back(s);
    } catch(runtime_er
      input.clear();
      string to_be_ign
      getline(input, t
    }
  }
  input.close();
}
```

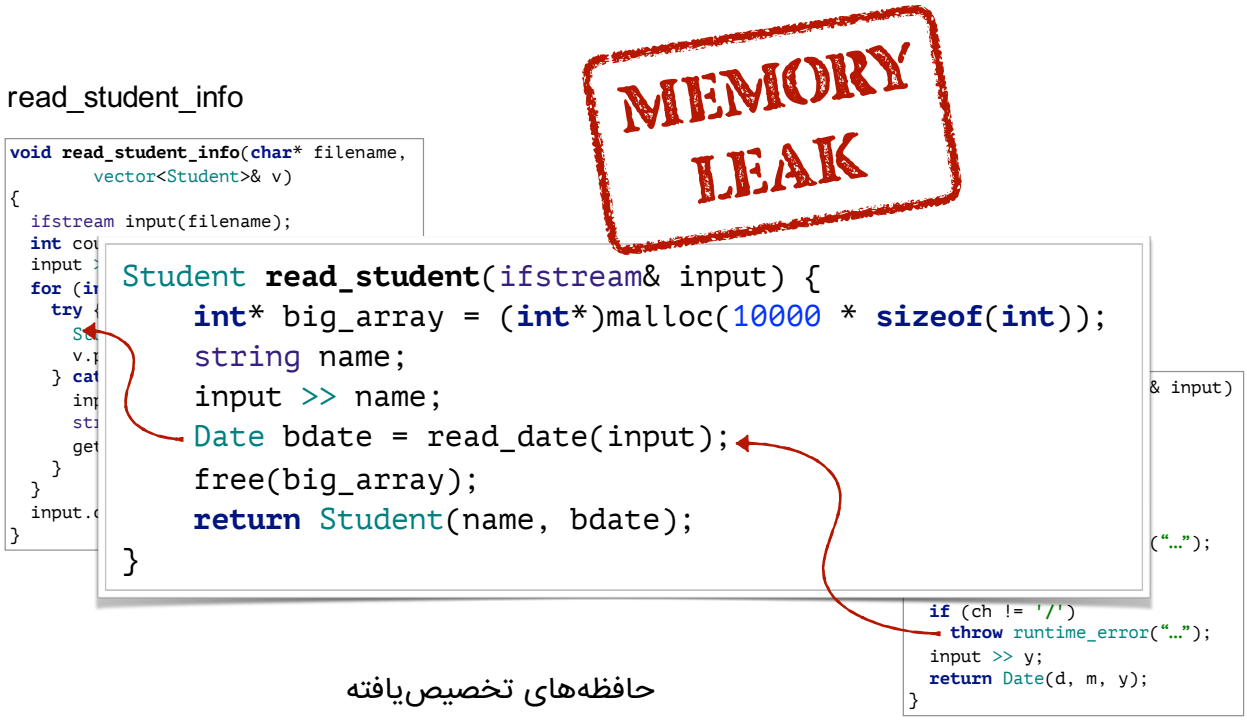
read_student

```
Student read_student(ifstream& input) {
    string name;
    input >> name;
    Date bdate = read_date(input);
    return Student(name, bdate);
}
```

ate

```
_date(ifstream& input)
m, y;
> d;
> ch;
!= '/')
  throw runtime_error("…");
input >> m;
input >> ch;
if (ch != '/')
  throw runtime_error("…");
input >> y;
return Date(d, m, y);
}
```

read_student_info



حافظه‌های تخصیص‌یافته

فایل‌های باز شده

اتصال‌های شبکه برقرار شده

... هر نوع منبع دیگری که نیاز به آزادسازی دارد

بازپرتاب (rethrow) استثناء

```
Student read_student(ifstream& input) {
    int* big_array = (int*)malloc(10000 * sizeof(int));
    try {
        string name;
        input >> name;
        Date bdate = read_date(input);
        free(big_array);
        return Student(name, bdate);
    } catch (...) {
        free(big_array);
        throw;
    }
}
```

اگر سازنده Student هم استثناء پرتاب کند دوبار free را فراخوانی خواهیم کرد.

```
Student read_student(ifstream& input) {
    int* big_array = (int*)malloc(10000 * sizeof(int));
    try {
        string name;
        input >> name;
        Date bdate = read_date(input);
        Student result(name, bdate);
        free(big_array);
        return result;
    } catch (...) {
        free(big_array);
        throw;
    }
}
```

راه اصولی‌تر، استفاده از الگوی RAII است که در آینده خواهیم دید.

RAII = Resource Acquisition Is Initialization

ترجمه استثناءها

```
void StudentDB::save_student(Student student) {
    try {
        // ... insert the student into the database
    } catch (DatabaseException& ex)
    if (ex.reason == DB_PK_VIOLATION)
        throw DuplicateStudentException(student.get_id());
    // handle other types of database exception
}
```

تمرین‌های کوتاه

۱. بعضی اوقات یک تابع استثنائی را می‌گیرد، اطلاعاتی به آن استثناء اضافه می‌کند و سپس آن را بازپرتاب می‌کند. آیا می‌توانید کاربردی برای این حالت ذکر نمایید؟

۲. با نوشتن یک برنامه کوچک تحقیق کنید آیا امکان داشتن بلوک‌های try/catch تودرتو داریم؟ در این حالت بازپرتاب از بلوک داخلی باعث چه چیزی می‌شود؟ آیا کاربردی برای استفاده از این بلوک‌های تودرتو متصور هستید؟

این سؤال را برای دو حالت پاسخ دهید: بلوک داخلی در قسمت try بلوک خارجی باشد یا در قسمت catch آن.