درس «مبانی کامپیوتر و برنامه‌سازی»

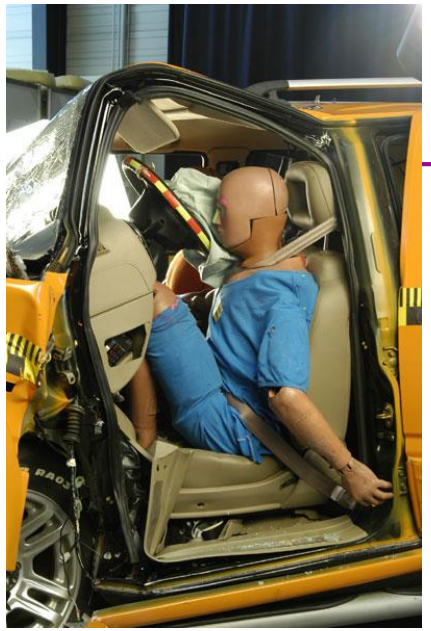
آزمایش برنامه، رفع اشکال و چند داستان دیگر

صادق علی‌اکبری

سرفصل مطالب

- آزمایش نرم‌افزار
- روش‌های رفع اشکال
  - Logging
  - Debugging
- استفاده از کامنت برای توصیف برنامه

اهمیت و لزوم آزمایش محصول



- تولیدکننده، قبل از تحویل محصول باید از کیفیت آن مطمئن شود
  - خودروساز، خودرو را قبل از تحویل به مشتری ارزیابی می‌کند
  - آشپز، قبل از مهمان غذا را می‌چشد
- کنترل کیفیت و تضمین کیفیت در صنایع مختلف مورد توجه است
- سهل‌انگاری در تست محصول، گاهی ویرانگر است

سهل‌انگاری در ارزیابی کیفیت نرم‌افزار

- موشک آریان۵ در سال ۱۹۹۶ توسط آژانس فضایی اروپا آزمایش شد
  - این موشک، ۴۰ ثانیه پس از پرتاب منفجر شد
  - هزینه: ۳۷۰ میلیون دلار
  - علت: وجود یک اشکال در نرم‌افزار
- سال ۱۹۸۰: پنج بیمار بر اثر دریافت مقدار زیاد پرتوی X جان باختند
  - علت: اشکال در نرم‌افزار ماشین پرتودرمانی

نیاز به آزمایش نرم‌افزار

- نرم‌افزار، مثل هر محصول دیگری، باید آزمایش شود
  - تا از کیفیت آن مطمئن شویم
- نرم افزاری که آزمایش نشده، هنوز کامل نیست
- ویژگی‌های یک نرم‌افزار خوب چیست؟
  - عملکرد صحیح
  - ویژگی‌های کیفی (غیرعملکردی)
    - کارایی، سرعت، سهولت استفاده، امنیت و غیره
- انواع آزمایش‌ها، کیفیت نرم‌افزار را از دیدگاههای مختلف می‌آزمایند

آزمایش نرم‌افزار

- ساده‌ترین شکل آزمایش: برنامه را به ازای ورودی‌های مختلف و شرایط متفاوت بیازماییم
- مثلاً اگر یک تابع می‌نویسیم، به ازای پارامترهای مختلف آن را امتحان کنیم
  - در تابع main به شکل‌های مختلف آن را فراخوانی کنیم ، مثال:

```
int main()
{
    cout << factorial(0);
    cout << factorial(1);
    cout << factorial(2);
    cout << factorial(5);
}
```

```
int main()
{
    cout << fibonacci(1);
    cout << fibonacci(2);
    cout << fibonacci(7);
}
```

- اگر نتایج مقبول نبود، باید برنامه را رفع اشکال کنیم
  - یعنی: ۱) محل دقیق اشکال برنامه را پیدا کنیم و ۲) اشکال را رفع کنیم

رفع اشکال (Debug)

- اشکالات نرم‌افزاری و اصطلاح باگ (Bug)
- مهمترین قدم در رفع اشکال: پیدا کردن اشکال
- دو رویکرد مهم برای پیدا کردن اشکال:
  - نمایش وضعیت برنامه در خروجی ← Logging
    - مثلاً با کمک cout مقدار متغیرها و مقادیر را در شرایط مختلف چاپ کنیم
    - و بعداً این خروجی‌ها را بخوانیم تا بفهمیم برنامه در چه شرایطی اشتباه کار کرده است
  - اجرای خط‌به‌خط برنامه ← معمولاً منظور از Debug کردن، این کار است
    - به جای اجرای کل برنامه، برنامه را خط‌به‌خط اجرا کنیم
    - در هر لحظه از اجرا، مقدار متغیرها و شرایط برنامه را بررسی کنیم

لاگ: مثال

```
#include <stdio.h>
int fib(int n) {
   if (n <= 0) return 0;
   if (n == 1) return 1;
   int a = 1, b = 1;
   for (int i = 3; i <= n; i++) {
      int c = a + b; a = b; b = c;
      printf("i=%d,a=%d,b=%d,c=%d\n", i,a,b,c);
   }
   return b;
}
int main() {
   int x ;
   scanf("%d", &x);
   printf("fib(%d)=%d", x,fib(x));
}
```

لاگ (log)

- بعد از رفع اشکال، باید همه لاگ‌ها را حذف کنیم
- نکته: روش‌های نوین Logging این کار را ساده می‌کنند

دیباگ

- محیط‌های توسعه نرم‌افزار (IDE) امکانات بسیار خوبی برای این کار دارند
  - مثل Eclipse و Visual-Studio
- مرور مفاهیم:

- Debug
- Breakpoint
- Steps (step into, step over, step return, run to line)
- Variable Values
- (Watch) Expression, Inspect
- Stack Trace

مرور و مثال در Eclipse

```
int recursive_factorial(int n) {
    if (n == 1)
        return 1;
    return n * factorial(n - 1);
}
int factorial(int n) {
    if (n < 0)
        return 0;
    int f = 1;
    while (n > 1)
        f *= n--;
    return f;
}
int main() {
    int n = 1;
    cout << factorial (n);
    n++;
    cout << factorial (n);
    cout << factorial (2*n+1);
    n=3;
    cout << recursive_factorial(2*n+1);
}
```

تکنیک Stackoverflow!

- وقتی به یک مشکل برمی‌خورید و راه‌حل آن را پیدا نمی‌کنید آن را در اینترنت جستجو کنید
- معمولاً افراد (بسیار) زیادی قبل از شما به این مشکل برخورده‌اند
- و افراد دیگری پاسخ مشکل را داده‌اند
- سایت‌هایی مثل stackoverflow.com در این زمینه معروف است

مفهوم کامنت (comment)

- توضیح یا کامنت: بخشی از برنامه که کامپایلر نادیده می‌گیرد
  - انگار در متن برنامه وجود ندارد
- کاربرد:
  - برای حذف (موقتی) بخشی از برنامه
  - کاربرد بهتر: توصیف و توضیح بخش‌هایی از برنامه
- کامنت یک خطی با // مشخص می‌شود
- کامنت چندخطی با */ شروع و با /* پایان می‌یابد

مثال

```
//This function returns n'th fibonacci number
int fib(int n) ;
/**
 * This is a recursive implementation of factorial function
 * parameter n is the input
 * returns n!
 */
int rfactorial(int n) ;
/**
* این تابع فاکتوریل است
* فاکتوریل مقدار پارامتر را برمیگرداند
 */
int factorial(int n) ;
int main() {
//printf("fib(%d)=%d", x,fib(x));
int n = 1;
cout << factorial(n) << endl;
...
}
```

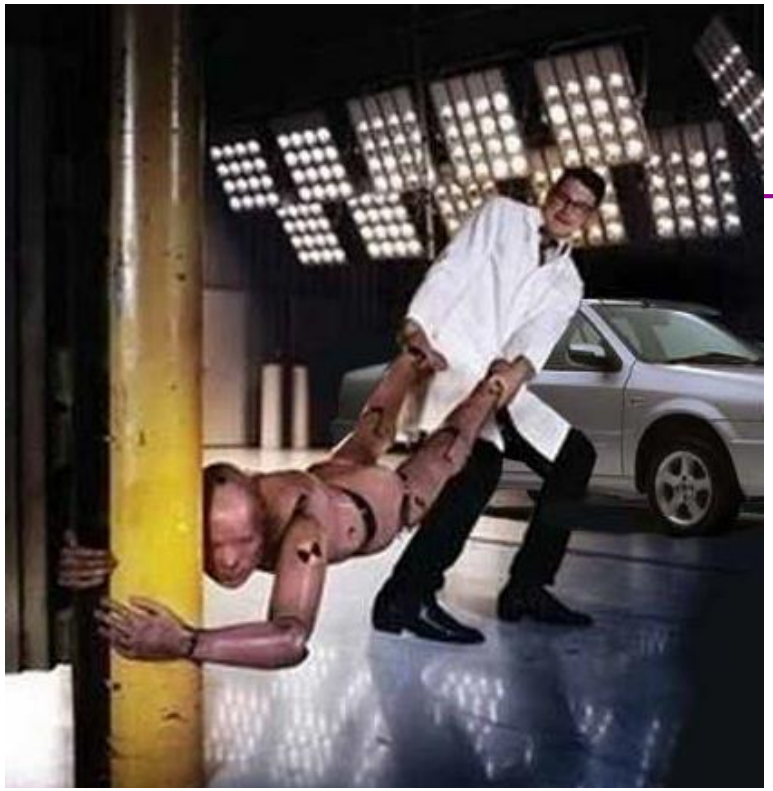
کیفیت برنامه

- برنامه‌نویسی ساخت‌یافته
  - مثلاً تقسیم‌بندی مناسب برنامه (به چند تابع)، عدم استفاده از goto و ...
- تمیز، حرفه‌ای و شیک بودن کد (دست‌خط برنامه‌نویس)
  - نامگذاری مناسب و گویا
  - دندانه‌گذاری مناسب
  - سایر استانداردها و رسوم برنامه‌نویسی (coding conventions)
- تست مناسب و کافی
- توصیف برنامه و مستندسازی مناسب با کمک کامنت

جمع‌بندی

جمع‌بندی

- اهمیت و جایگاه آزمایش نرم‌افزار
- روش‌های تست نرم‌افزار
- رفع اشکال برنامه و روش‌های آن
  - Logging
  - Debugging
- مستندسازی برنامه با کمک کامنت



جستجوی بیشتر

- فرصت‌ها و تهدیدها در تست نرم‌افزار
  - در مقایسه با تست سایر محصولات مهندسی (خودرو، ساختمان و ...)
- مفهوم Unit Testing
  - CppUnit, GoogleTest, ...
- مفهوم Log و ابزارها و الگوهای این حوزه
- تفاوت clog و cerr با cout
  - cout << "Console output" << endl;
  - cerr << "Console error" << endl;
  - clog << "Console error type 2" << endl;
- سایر امکانات مهم محیط‌های توسعه (IDE) ، مثل Refactoring
- تولید اتوماتیک مستندات از روی کامنت‌ها چگونه ممکن است (مثلاً Doxygen)

پایان