Testes

Prof. Dr. Fabio Kon

Departamento de Ciência da Computação
IME / USP

Lab. Prog. I - 2010

(Com raras alterações introduzidas em 2013 por Kelly Rosa Braghetto)

Testar ≠ Depurar

- Simplificando
  - Depurar - o que se faz quando se sabe que o programa não funciona;
  - Teste - tentativas sistemáticas de encontrar erros em programa que você “acha” que está funcionando.

- “Testes podem mostrar a presença de erros, não a sua ausência (Dijkstra)”

Teste enquanto você escreve código

- Se possível, escreva os testes antes mesmo de escrever o código
  - uma das técnicas de XP
- quanto antes for encontrado o erro melhor !!

Técnicas básicas

(Conselhos simples e úteis!)

- Teste o código em seus limites;
- Teste pré e pós condições;
- Use premissas (*assert*);
- Programe defensivamente;
- Verifique os códigos de erro.

Teste o código em seus limites

- Verifique o funcionamento de cada pequeno trecho de código (um laço ou if, por exemplo)
  - o laço é executado o número correto de vezes?
  - o desvio do fluxo de execução vai para o “ramo” correto?
- Teste uma entrada vazia, um vetor com um único item, um vetor cheio, etc.

Exemplo

Um trecho que lê caracteres até que o '\n' seja encontrado ou até que o buffer seja completamente preenchido

```
int i;
char s[MAX];

for(i=0; s[i] = getchar() != ‘\n’ &&
            i < MAX - 1; i++);
s[--i]=‘\0’;
```

Problemas:

- precedência do '=' é menor do que a do “!=”
- não funciona para linha vazia

Exemplo (continuação)

```
int i;
char s[MAX];

for(i=0; i < MAX - 1; i++)
    if ((s[i] = getchar()) == '\n')
        break;
s[i]='\0';
```

Testes:

- linha vazia ok; 1 caractere ok; 2 caracteres ok; MAX caracteres ok

Problema: e se o primeiro caractere já é o de fim de arquivo?

Exemplo (continuação)

```
int i;
char s[MAX];

for(i=0; i < MAX - 1; i++)
    if (s[i] = getchar()) == ‘\n’ || s[i]==EOF)
        break;
s[i]=‘\0’;
```

Testes:

- tudo ok.

Problemas (?):

- O que se deve fazer se a string s fica cheia antes do ‘\n’?

[depende do fato desses caracteres serem necessários ou não]

Teste de pré e pós-condições

- Verificar certas propriedades antes e depois de trechos de código

```
double avg(double a[], int n){
    int i;
    double sum = 0.0;

    for(i = 0; i < n; i++)
        sum += a[i];
    return sum / n;
}
```

Problema:

- se n == 0, ocorre uma divisão por zero.

Teste de pré e pós condições

- Solução possível

```
// mudar o return
return n <= 0 ? 0.0 : sum / n;
```

- Não existe uma única resposta certa
  - A única resposta claramente errada é ignorar o erro !!
  - Ex.: USS Yorktown, em 1998 (navio de guerra, cruzador de mísseis)
    - Uma divisão por zero desligou o sistema propulsão

Uso de premissas

- Em C e C++, use <assert.h>
  - ex: `assert (n>0);`
  - se a premissa for violada, o programa é abortado:

    ```
    Assertion failed: n>0, file
      avgtest.c, line 7.
    ```

- Ajuda a identificar “culpados” pelos erros

Programação defensiva

- Tratar situações que não “deveriam” acontecer. Exemplo:

```
if (nota < 0 || nota > 10) // não pode acontecer
    letra = ‘?’;
else if (nota > 9)
    letra = ‘A’;
else ...
```

Utilizar códigos de erro

- Checar os códigos de erro de funções e métodos
  - você sabia que o scanf retorna o número de parâmetros lidos, ou EOF ?
- Sempre verificar se ocorreram erros ao abrir, ler, escrever e, principalmente, fechar arquivos
- Em Java, sempre tratar as possíveis exceções

Pequeno exercício:

```
int fatorial(int n) {
    int fat = 1;

    while (n--) {
        fat *= n;
    }
    return fat;
}
```

como testar isso?

Testes sistemáticos (1/4)

- **Teste incrementalmente**
  - durante a construção do sistema
    - após testar dois pacotes independentemente teste se eles funcionam juntos
- **Teste primeiro partes simples**
  - tenha certeza que partes básicas funcionam antes de prosseguir
  - testes simples encontram erros simples
  - teste as funções/métodos individualmente
    - Ex: teste de função que faz a busca binária em inteiros

Testes Sistemáticos (2/4)

- **Conheça as saídas esperadas**
  - conheça a resposta certa
  - para programas mais complexos valide a saída com exemplos conhecidos
    - compiladores - arquivos de teste;
    - numéricos - exemplos conhecidos, características;
    - gráficos - exemplos, não confie apenas nos seus olhos.

Testes Sistemáticos (3/4)

- **Verifique as propriedades invariantes**
  - alguns programas mantém propriedades da entrada
    - número de linhas
    - tamanho da entrada
    - frequência de caracteres
      - Ex: a qualquer instante o número de elementos em uma estrutura de dados deve ser igual ao número de inserções menos o número de remoções.

Testes Sistemáticos (4/4)

- **Compare implementações independentes**
  - os resultados devem ser os mesmos
    - se forem diferentes pelo menos uma das implementações está incorreta
- **Meça a cobertura dos testes**
  - cada comando do programa deve ser executado por algum teste
    - existem *profilers* que indicam a cobertura de testes

Automação de Testes

- Testes manuais
  - tedioso, não confiável
- Testes automatizados
  - devem ser facilmente executáveis
    - junte em um *script* todos os testes

Automação de Testes

- **Automatize os testes de regressão**
  - Comparar a nova versão com a antiga
  - verificar se os erros da versão antiga foram corrigidos
  - verificar que novos erros não foram criados

- Testes devem rodar de maneira silenciosa
  - se tudo estiver OK

Automação de Testes

Exemplo de script que automatiza um teste de regressão de um programa chamado *killer application* (ka) :

```
for i in ka_data.*      # laço sobre os testes
do
  old_ka $i > out1      # versao antiga
  new_ka $i > out2      # nova versao
  if !cmp -s out1 out2 # compara
  then
    echo $i: Erro       # imprime mensagem
  fi
done
```

Automação de Testes

- **Crie testes autocontidos**
  - testes que contêm suas próprias entradas e respectivas saídas esperadas
- O que fazer quando um erro é encontrado?
  - se não foi encontrado por um teste, faça um teste que o provoque
- Nunca se desfaça de um teste
  - mantenha um registro de erros, mudanças e consertos

Ambiente de Testes

- Às vezes, para se testar um componente isoladamente é necessário criar um ambiente com características de onde este componente será executado
  - ex: testar funções mem* do C (como memset)

Ambiente de testes

```
/* memset: set the first n bytes of s to the byte c */
void *memset(void *s, int c, size_t n) {
    size_t i;
    char *p;

    p = (char *) s;
    for (i=0; i<n; i++)
        p[i] = c;
    return s;
}

// slow memset(s0 + offset, c, n);
// fast memset2(s1 + offset, c, n);
// compare s0 e s1 byte a byte
```

Como testar funções do math.h ?

Testes de Estresse

- Testar com grandes quantidades de dados
  - gerados automaticamente
  - erros comuns:
    - *overflow* nos buffers de entrada, vetores e contadores
  - Exemplo: ataques de segurança
    - gets do C - não limita o tamanho da entrada
    - o scanf(``%s'', str) também não...
    - Erro conhecido por "buffer overflow error"
    - Opções mais seguras:
      - ***fgets(buf, sizeof(buf), stdin)***
      - ***scanf(``%20s'', buf)***

Testes de Estresse

Exemplos de erros que podem ser encontrados:

```
char *p;

p = (char *) malloc (x * y * z);
```

- Problemas de conversão entre tipos diferentes causaram a explosão do foguete Ariane 5 (1996)
  >>> conversão de double de 64 bits em int de 16 bits => BOOM

Dicas para Fazer Testes

- Cheque os limites dos vetores
  - (caso a linguagem não faça isso por você)
  - Para os testes, faça com que o tamanho dos vetores seja pequeno (ao invés de criar entradas para testes muito grandes)
- Faça funções de hashing que devolvem constantes
  - para avaliar se o encadeamento está funcionando
- Crie versões de malloc que ocasionalmente falham
  - para testar se o seu código se recupera de erros do tipo “out-of-memory”

Dicas para Fazer Testes

- Inicialize os vetores e variáveis com um valor não nulo
  - ex: 0xDEADBEEF pode ser facilmente encontrado olhando no depurador
- Não continue a implementação de novas características se já foram encontrados erros
- Teste em várias máquinas, compiladores e SOs
- Desligue todos os testes antes de lançar a versão final

Lembre-se

- Por que não escrever testes ?
  - estou com pressa!
- Quanto maior a pressão,...
  - ... menos testes
- Com menos testes, ...
  - ... menos produtividade e menor estabilidade
- Logo, a pressão aumenta....

O único conceito mais importante de testes é

DO IT

Baseado em

- Baseado em:
  - **Livro:** *The Practice of Programming*
    Kernighan & Pie
    (Capítulo 6)

Para ver mais sobre Testes...

- Veja a versão completa desta apresentação (que inclui também *Teste de Software Orientado a Objetos* e *Teste em Métodos Ágeis*). Disponível em:

  http://www.ime.usp.br/~kon/presentations/testes2010.odp

- Leia o artigo *"A Importância dos Testes Automatizados"*.

  Disponível em:

  http://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf