Depuração

Samuel Oliveira

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

IMD
INSTITUTO METRÓPOLE DIGITAL

Contexto

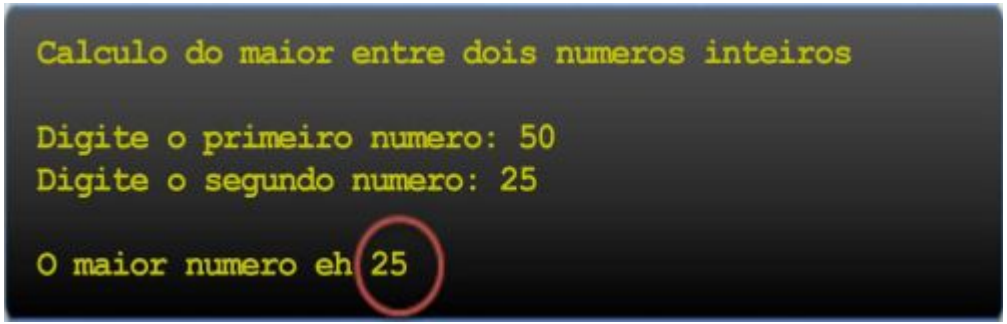
- Quando programamos, geralmente inserimos erros (bugs) de diversas formas nos programas;
- Desde os mais básicos (sintáticos ou estruturas) que levam ao compilador a gerar uma mensagem de erro;
- Aos erros mais semânticos inseridos na lógica do programa, para os quais uma simples leitura de código pode não ser suficiente;
- Nesse último caso, dois procedimentos importantes são geralmente praticados para tentar encontrar uma solução:
  - Testes: para descobrir o erro;
  - Depuração: para localizar e eliminar o erro;

Teste de Software

- Procedimento de detecção de erros em um programa;
- Bastante popular no que se refere a melhoria da qualidade;
- Testes podem ser realizados por:
  - Desenvolvedores de software (teste de caixa branca);
  - Especialistas no domínio de software (testes de caixa preta);
- São classificados em diversas categorias:
  - Teste unitário;
  - Teste de funcional;
  - Teste de integração;
  - Teste de regressão;

Exemplo de Teste 1

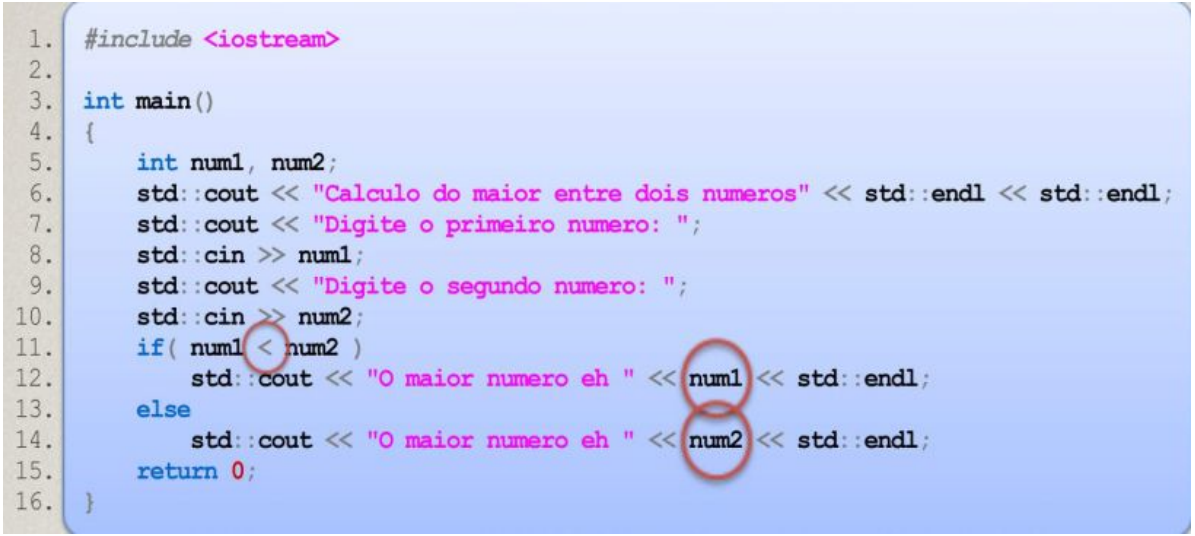
- Execução de um programa que lê dois números e imprime o maior deles



- Podemos detectar um erro no programa

Exemplo de Teste 2

- Programa que recebe de entrada dois números e imprime o maior deles



```
1.  #include <iostream>
2.
3.  int main()
4.  {
5.      int num1, num2;
6.      std::cout << "Calculo do maior entre dois numeros" << std::endl << std::endl;
7.      std::cout << "Digite o primeiro numero: ";
8.      std::cin >> num1;
9.      std::cout << "Digite o segundo numero: ";
10.     std::cin >> num2;
11.     if( num1 < num2 )
12.         std::cout << "O maior numero eh " << num1 << std::endl;
13.     else
14.         std::cout << "O maior numero eh " << num2 << std::endl;
15.     return 0;
16. }
```

- A impressão do maior é efetuada incorretamente
- Ou o sinal de comparação está invertido

Caso de Teste

- É um conjunto de valores constituídos por pares de entrada e saída esperada;
  - <entradas; **saídas esperadas**>
- Exemplos de casos de teste para um programa que calcula o maior número entre dois inteiros:
  - <100, 20; **100**>
  - <49, 87; **87**>
  - <-85, 37; **37**>
  - <0, -10; **0**>
  - <-15, ‘a’; **entrada inválida**>
  - <”lp1, ‘a’; **entrada inválida**>

Dificuldades

- Encontrar erros em programas pode se tornar uma tarefa difícil e exigir muito tempo;
- Portanto, todo programador deveria procurar:
  - Escrever um código de boa qualidade;
  - Estudar e aplicar técnicas que ajudam a evitar erros;
  - Estar ciente das características das linguagens utilizadas;
  - Escrever bons casos de teste;
  - Usar boas ferramentas de depuração;

Depuração

- *Debug* ou *debugging* é um procedimento para diagnóstico e correção de erros já detectados em um programa;
- É muito útil, pois permite ao programador:
  - Monitorar a execução de um programa (passo a passo);
  - Ativar pontos de parada (linha, função, condição);
  - Monitorar os valores das variáveis;
  - Alterar áreas da memória;
  - Monitorar as chamadas das funções;
  - Retroceder na execução do programa;

Depuração

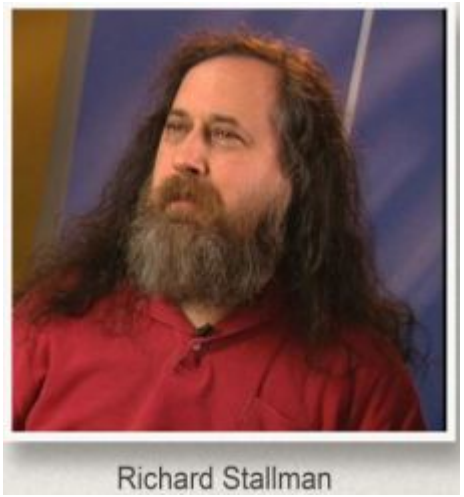
- Em geral, mais da metade do tempo gasto no desenvolvimento de um software é gasto com a depuração;
- Apesar da boa utilidade, a depuração não é sempre a melhor alternativa;
- Certos tipos de programas não se dão muito bem com ela:
  - Sistemas operacionais, sistemas distribuídos, múltiplos threads, sistemas de tempo real;
- Não é possível em algumas linguagens e ambientes;
- Pode variar muito de um ambiente para o outro;
- Pode ser complicada para programadores iniciantes;

Depuração

- Uma solução alternativa à depuração se passa pelo uso criterioso dos comandos de impressão (printf, std::cout);
- Estrutura de mensagens que mostra por onde o código está passando;
- Mas, o código original é alterado para poder inserir estas mensagens (ruim);
- Um grande interesse da depuração é que ela permite mostrar por onde o código está passando sem alterar o código original (bom);

Depurador *GNU Debugger* (GDB)

- GDB é um depurador do GNU que suporta diversas linguagens de programação:
  - C, C++, Objective-C, Java, Fortran, etc;
- Foi criado por Richard Stallman em 1986;
- É mantido pelo comitê GDB nomeado pela Free Software Foundation;
- Tal qual os compiladores GCC e G++ do GNU, o depurador GDB também pode ser integrado a diversos ambientes de desenvolvimento;
  - Netbeans, Eclipse, Code::Blocks, Dev-C++, Visual Studio Code, etc;


Richard Stallman

Exemplo de Código para Depuração

- Programa que recebe como entrada a nota de alunos e calcula a média;

```cpp
#include <iostream>

using namespace std;

int main(){
    int num_alunos;

    cout << "Quantos anos tem na turma?" << endl;
    cin >> num_alunos;

    double notas[num_alunos];
    double media = 0;

    for(int i = 0; i < num_alunos; i++){
        cout << "Digite a nota do aluno " << i + 1 << ":" << endl;
        cin >> notas[i];
        media += notas[i];
    }

    cout << "A média calculada é: " << media/num_alunos << endl;

    return 0;
}
```

Depuração com Terminal de Comandos

- As diretivas do compilador permitem:
  - Inserir informações de depuração no código do programa (-g);
  - Indicar ao compilador que toda forma de otimização deve ser eliminada a fim de facilitar o funcionamento do depurador (-O0);
- Uma vez a execução do depurador iniciada, podemos utilizar diversas opções fundamentais para a depuração do código:
  - r: executa o programa do início;
  - q: sai do GDB;
  - b número: insere um ponto de parada na linha número;
  - d número: apaga o ponto de parada da linha número;
  - p variável: imprime o valor atual da variável;
  - c: continua a execução do programa sem paradas;
  - k: força o término da execução do programa;

Depuração com Terminal de Comandos

```
Compilação: g++ -g -O0 media.cpp -o programa
Execução:   gdb programa
```

Depuração

Samuel Oliveira

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

IMD
INSTITUTO METRÓPOLE DIGITAL