Programação

IFMG CODAAUT

Prof. Marco Antonio M. Carvalho

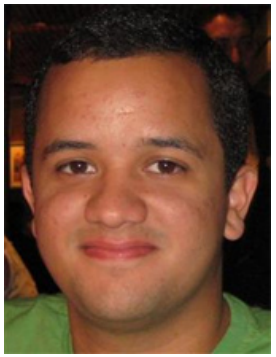
UFOP
Universidade Federal de Ouro Preto

PROEX
Pró-reitoria de extensão da UFOP

decom
departamento de computação

INSTITUTO FEDERAL
MINAS GERAIS

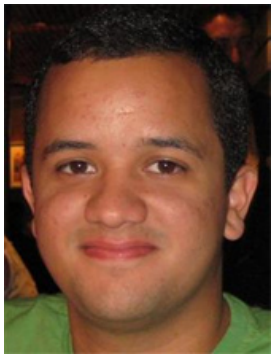
Quem Sou Eu?



- Bacharel em Ciência da Computação (2005) – Faculdades Integradas de Caratinga;
- Mestre em Engenharia Eletrônica e Computação (2008) – ITA;
- Doutor em Engenharia Eletrônica e Computação (2013) – ITA;
- Interesses:
  - Teoria da Computação, Otimização Combinatória, Pesquisa Operacional;
  - Análise de Risco;
  - **Maratona de Programação**.
- Professor do DECOM/UFOP desde 2010/2
  - Disciplinas de programação;
  - Maratona de Programação;
  - Membro do GOAL-UFOP

GOAL
Group of Optimization and Algorithms

Quem Sou Eu?



- Competidor:
  - Olimpíada Brasileira de Informática 2002;
  - Fase Sulamericana da Maratona de Programação 2003;
  - Fase Regional Maratona de Programação 2004.
- Técnico
  - Maratona de Programação 2010, 2012 e 2013;
  - Maratona Mineira de Programação 2012 e 2013;
  - Maratona Mineira Online 2012;
  - + Seletivas UDESC, USP, etc.
- Organização
  - Maratonas DECOM 2012 e 2013
- Coordenação
  - Projeto FAPEMIG "Treinamento para Maratona de Programação" 2011-2013.

Contato

- marco.opt@gmail.com
- 3559-1663
- Sala COM45 – DECOM/ICEB III

Programa Ação

Programa Ação

- Parceria entre DECOM/CODAAUT, UFOP/IFMG para ensino de programação e algoritmos avançados;
- Dois projetos de extensão
  - UFOP/PROEX;
  - IFMG.
- **Objetivo**: Capacitar alunos do ensino médio em habilidades de programação;
- **Métrica**: Olimpíada Brasileira de Informática

Programa Ação

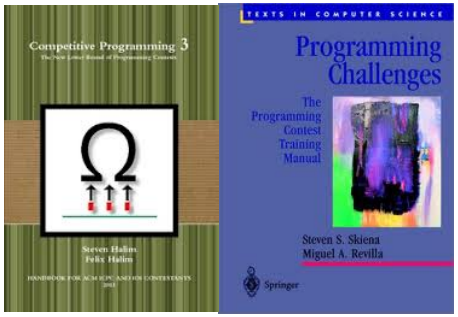
- Uma sessão de treinamento semanal por turma
  - Laboratório de ensino COM30;
  - Turma AUT1: Quintas 15:30-16:45;
  - Turma AUT2: Sextas 15:30-16:45.

- **Assinatura do termo de compromisso para utilização do laboratório.**

Recursos

Recursos

- **Lista de discussão**
  - programaacao@googlegroups.com
  - Solicitem acesso: http://groups.google.com/group/programaacao
- **Página com material dos treinamentos**
  - http://www.decom.ufop.br/marco/extensao/obi/
- **Repositório online de problemas das edições passadas da OBI**
  - Juiz automático;
  - SPOJ;
  - http://br.spoj.com/problems/obi/sort=-7.
- **Bibliografia sobre competições de programação**



Recursos (cont.)

- **Bolsas BIC Jr.**
  - Bolsas para alunos do ensino médio que desempenhem atividades da UFOP;
  - R$100,00;
  - Abertura do edital em outubro.

Metodologia

Metodologia

- As sessões de treinamento são realizadas em laboratório
  - Aula expositiva;
  - Contextualização de problemas computacionais e práticos do dia-a-dia;
  - Relação entre descrição textual de problemas e aspectos computacionais
    - Algoritmos e estruturas de dados.
  - Seleção de problemas no SPOJ.

Metodologia

- O material da aula utiliza a linguagem **C++**
  - Linguagem mais utilizada em competições de programação.
- O ambiente computacional será *Linux*
  - Integração com ambientes de programação;
  - Facilidade de compilação por linha de comando.

Metodologia

- Não será cobrado a memorização de algoritmos e estruturas
  - Ao invés disso, serão cobrados **quando** e **como** utilizá-los.
- O objetivo reside em aprender a utilizar as ferramentas disponíveis, e não em fabricar as ferramentas.

O Quê se Espera do Aluno

O Que se Espera do Aluno?

- Pontualidade
  - Chamada.
- Dedicação exclusiva às atividades da disciplina durante a aula;
- Proatividade;
- Aplicação nas atividades extra-classe
  - *Não se aprende a andar de bicicleta apenas vendo outras pessoas pedalando*.

O Que se Espera do Aluno?

- Haverá grande flexibilidade no ambiente da aula no que diz respeito ao compartilhamento de conhecimento
  - O que não inclui cópias.
- A idéia é que os alunos interajam fortemente, troquem experiências e se ajudem
  - O desempenho é avaliado individualmente, mas o aprendizado deve ser coletivo.
- Pode ser utilizada a lista de e-mails para isso, ou mesmo o contato pessoal.
- Todas as dificuldades e dúvidas devem ser postas em discussão
  - Todos contribuem;
  - Todos aprendem.

Olimpíada Brasileira de Informática

Olimpíada Brasileira de Informática

Modalidade

*"2.2 Na Modalidade Programação, as tarefas da prova versarão sobre problemas de programação, de **dificuldade média**, exigindo **conhecimento de estruturas de dados e técnicas de programação**."*

Eligibilidade - Modalidade Programação - Nível 2

*"3.6 É permitida a participação de alunos que estejam cursando o Ensino Fundamental ou **Ensino Médio**, ou que tenham encerrado o Ensino Médio em dezembro do ano anterior à sua participação na OBI."*

Olimpíada Brasileira de Informática

- **Linguagens de Programação**
  - As únicas linguagens de programação aceitas nas duas fases (incluindo a seletiva para Olimpíada Internacional) são:
    - C;
    - **C++**;
    - Pascal.

Olimpíada Brasileira de Informática

- **Provas**

  *"5.9 Durante a prova, cada participante deverá ter acesso **individual** a um computador pessoal, com capacidade adequada de processamento, **sem acesso à Internet**, com um ambiente de programação (no mínimo um editor de texto) e compiladores para as linguagens de programação permitidas pela OBI (ao menos uma linguagem)."*

Como Ser Competitivo

(Adaptado da *National University of Singapore School of Computing*)

Como Ser Competitivo

Dica 1: Digite Rápido e Corretamente

- Não é brincadeira, é importante!
- Teste a si próprio
  - http://www.typingtest.com
  - ZEBRA – Africa's stripped horse
- Familiarize-se com a posição das teclas
  - (,), {, }, [,], <, >, ', ", &, | , !, etc;
  - Domine o *autocomplete* de algum editor.

Como Ser Competitivo

Dica 2: Identifique Rapidamente o Tipo do Problema

- *Ad Hoc* (algoritmos não tradicionais);
- Busca Completa;
- Dividir e Conquistar;
- Guloso;
- Programação Dinâmica;
- Grafos;
- Matemático;
- *String*;
- Geometria Computacional;
- Outros mais difíceis.

Como Ser Competitivo

Dica 3: Analise o Algoritmo

- Veremos o básico necessário
  - Localizar as restrições no enunciado do problema;
  - Pensar no algoritmo mais simples que funcione;
  - Realizar análises básicas que convençam que o algoritmo funciona
    - Antes de começar a codificar!

Como Ser Competitivo

Dica 4: Domine uma Linguagem de Programação

- Devemos dominar pelo menos uma linguagem de programação
  - Menos tempo olhando em referências;
  - Usar atalhos, macros, etc;
  - Usar bibliotecas sempre que possível.
- A idéia é, uma vez com a solução em mente, traduzí-la em um código livre de erros
  - E rápido.

Como Ser Competitivo

Dica 5: Dominar a Arte de Testar

- Obviamente, queremos um *Código Aceito!*
  - Nossos códigos têm que passar pelos "testes secretos" dos juízes.
- Entretanto, nem sempre é possível
  - Onde foi que eu errei?

Como Ser Competitivo

Dica 6: Prática e Mais Prática

Os Problemas

Os Problemas

- Os problemas são enunciados de forma bem humorada, em contextos fictícios, porém, de aplicação prática;
- Envolvem, dentre outros:
  - Aritmética e Álgebra;
  - Geometria computacional;
  - Manipulação de *strings*;
  - Grafos;
  - Problemas Combinatórios.

30

O Primeiro Problema

- Ver o problema 3n+1.

SPOJ – *Sphere Online Judge*

SPOJ – *Sphere Online Judge*

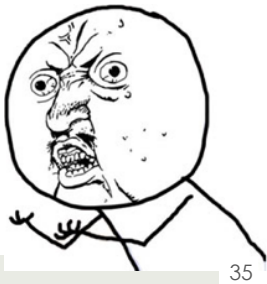
- O SPOJ é um repositório de problemas de programação
  - Possui uma versão em português;
  - Possui uma seção dedicada para problemas da OBI;
  - O programador seleciona um problema, envia o código-fonte e recebe uma mensagem
    - Erro; ou
    - Código Aceito!
- Cadastrem-se no SPOJ, colocando **[IFMG/UFOP]** como instituição
  - Acompanhamento do desempenho dos alunos pelo ranking do site.

Tipos de Erros

- O *juiz* exibe uma mensagem após a correção do código-fonte enviado;
- No caso de erro, uma mensagem específica é exibida para que o código-fonte seja corrigido e submetido novamente
  - No entanto, a mensagem nem sempre é específica sobre a localização do erro.

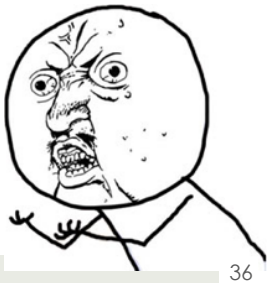
Tipos de Erros

- **Resposta errada**
  - Novamente, a bateria de testes é extensa, e embora seu programa tenha executado normalmente para os testes que você fez, há algo de errado;
  - Realize testes diferentes;
  - Verifique também os limites dados no programa, por exemplo:
  - "Leia um inteiro *n* e imprima n^2". Se *n* for 100000 e você estiver lendo *n* como inteiro e imprimindo n^2com %d vai ocorrer *overflow* na variável e provavelmente vai dar resposta errada.



Tipos de Erros

- **Resposta errada (cont.)**
  - A resposta dada pode estar em formato errado em relação ao que foi pedido;
  - Uma diferença mínima, como uma quebra de linha já é suficiente;
  - Note que, neste caso, não foi avaliado se a solução está certa ou errada.



Tipos de Erros

- **Erro em tempo de execução**
  - O programa deu pau em algum dos testes realizados;
  - Note que a bateria de testes é extensa, e embora seu programa tenha executado normalmente para os testes que você fez, há algo de errado;
  - Procure por erros de memória.

Erro em Tempo de Execução

- Não se esqueça do *return 0*!

```
int main (void) {
    //seucodigo
    return 0;
}
```

Erro em Tempo de Execução

- Não faça divisões por zero

```
int main () {
    while (n>=0) {
        /* Note que quando n==0, você fará uma divisão por zero*/
        printf("%0.2f\n",10/n);
        n--;
    }
    return 0;
}
```

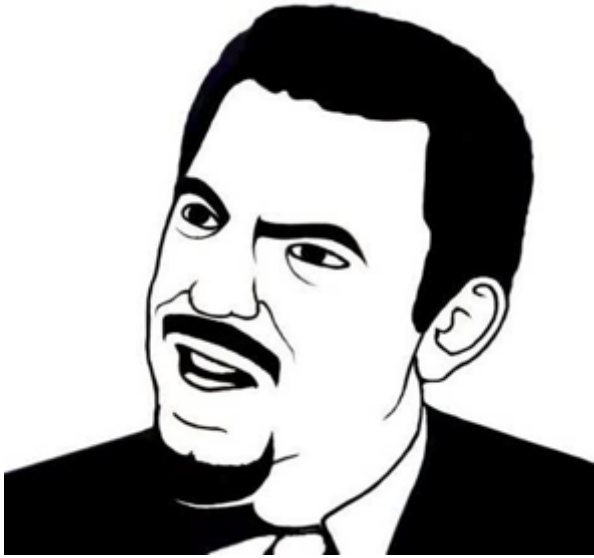
Erro em Tempo de Execução

- Não acesse memória que não lhe pertence

```
int main (void) {

   int v[100];

   //a posição 101 não te pertence

   printf("O elemento na posição 101 é: %d\n", v[101]);

   return 0;

}
```

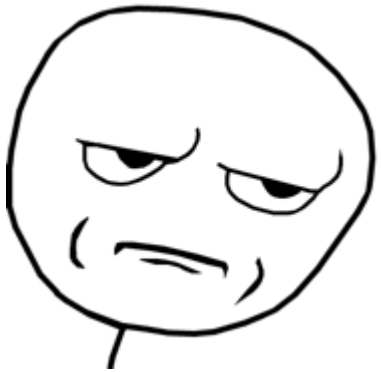
Tipos de Erros

- **Erro de compilação**
  - Simples assim.



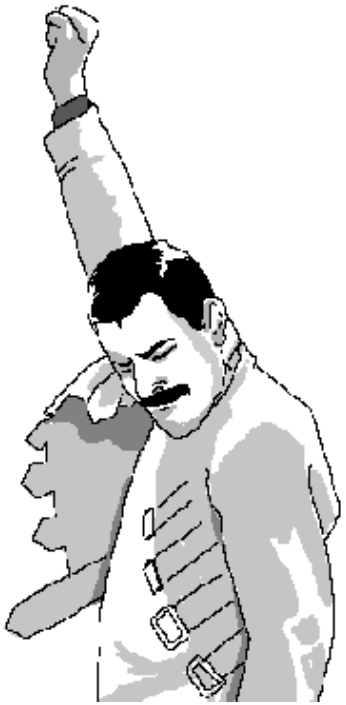
Tipos de Erros

- **Tempo limite excedido**
  - O programa demorou demais a responder;
  - Não foi avaliado se a resposta está certa ou errada;
  - Procure por procedimentos "pesados" ou *loops* infinitos.



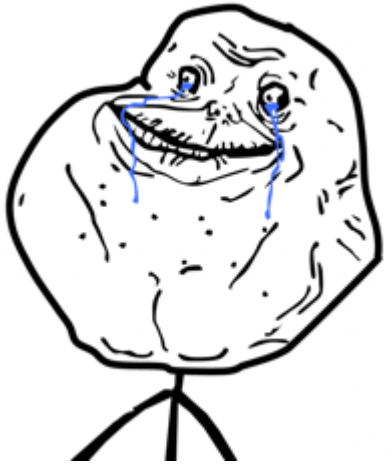
Código Aceito

- Código aceito!



O que não é permitido

- Internet;
- Aparelhos eletrônicos;
- Comunicação externa;
- Chamadas de sistema.



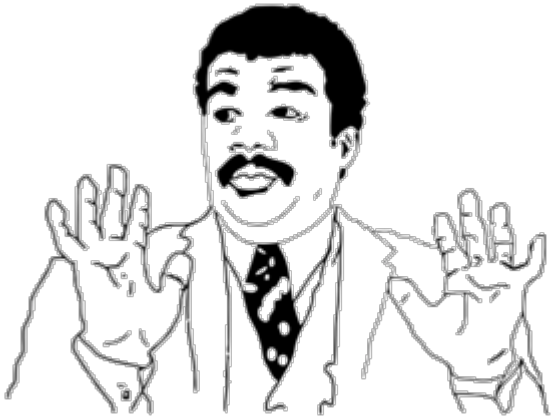
Dicas Úteis

Exemplo de Código Padrão

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <cctype>
#include <algorithm>
#include <map>
#include <queue>
#include <stack>
#include <vector>
#include <iostream>
using namespace std;

const int INF = 0x3F3F3F3F;
const int NULO = -1;
const double EPS = 1e-10;

_inline(int cmp)(double x, double y = 0, double tol = EPS) {
   return (x <= y + tol) ? (x + tol < y) ? -1 : 0 : 1;
}

int main(){
   int i, j;

   return 0;
}
```

Overflow e Final de Linha

- Fique atento ao intervalo dos valores da entrada e saída
  - *Overflow* aritmético.
- Não esquecer do return 0 ao final;
- Toda linha termina com '**\n**'
  - Causa *Resposta Errada*.



Entrada e Saída

- Em todos os problemas que veremos, os dados são lidos da entrada padrão e escritos na saída padrão
  - Nada de abrir arquivos ou chamadas de sistema.
- Nos testes, utilizaremos redirecionamento da entrada
  - ./programa < entrada.txt

Leitura de Dados

- Existe um aspecto importante sobre a leitura de dados
  - Até quando ler?
- Diz respeito ao término da entrada
  - Número fixo de valores, valor especial ou EOF.
- O enunciado do problema deve ser lido atentamente
  - A codificação da leitura dos dados não deve ser um processo lento.

Problemas Sugeridos

- Nível 0;
- Problemas OBI;
- **Ver no site.**

Perguntas?