C/C++

Crash Course C/C++

Filosofia da Programação
1° semestre de 2004

Árvore Genealógica

- Out 1956 - FORTRAN I

Árvore Genealógica

- Out 1956 - FORTRAN I
- Fev 1958 - IAL

Árvore Genealógica

- Out 1956 - FORTRAN I
- Fev 1958 - IAL
- 1958 - ALGOL (58)

Árvore Genealógica

- Out 1956 - FORTRAN I
- Fev 1958 - IAL
- 1958 - ALGOL (58)
- 1960 - ALGOL (60)

Árvore Genealógica

- Out 1956 - FORTRAN I
- Fev 1958 - IAL
- 1958 - ALGOL (58)
- 1960 - ALGOL (60)
- 1963 - CPL

Árvore Genealógica

- Out 1956 - FORTRAN I
- Fev 1958 - IAL
- 1958 - ALGOL (58)
- 1960 - ALGOL (60)
- 1963 - CPL
- Jul 1967 - BCPL

Árvore Genealógica

- 1969 - B

Árvore Genealógica

- 1969 - B
- 1971 - C

Árvore Genealógica

- 1969 - B
- 1971 - C
- 1980 - C + Classes

Árvore Genealógica

- 1969 - B
- 1971 - C
- 1980 - C + Classes
- Jul 1983 - C++ (branch Simula 67)

Árvore Genealógica

- 1969 - B
- 1971 - C
- 1980 - C + Classes
- Jul 1983 - C++ (branch Simula 67)
- 1989 - ANSI C

Árvore Genealógica

- 1969 - B
- 1971 - C
- 1980 - C + Classes
- Jul 1983 - C++ (branch Simula 67)
- 1989 - ANSI C
- 1998 - ANSI C++

Linguagem C

- High Level Language (objetos,módulos,etc).

Linguagem C

- High Level Language (objetos,módulos,etc).
- Low Level Language (registradores,hardware,etc).

Linguagem C

- High Level Language (objetos,módulos,etc).
- Low Level Language (registradores,hardware,etc).
- Alcança um vasto espectro de aplicações.

Linguagem C

- High Level Language (objetos,módulos,etc).
- Low Level Language (registradores,hardware,etc).
- Alcança um vasto espectro de aplicações.
- Obviamente não pode ser (a mais) adequada para tudo.

Morfologia

- Programação é Poesia

Morfologia

- Programação é Poesia
- Código é Texto Puro

Morfologia

- Programação é Poesia
- Código é Texto Puro
- O Texto é Puro, Claro e Honesto

Morfologia

- **métrica** padroniza o código

Morfologia

- **métrica** padroniza o código
- **visualização** torna-se prática então

Morfologia

- **métrica** padroniza o código
- **visualização** torna-se prática então
- **participação** trabalho em grupo torna-se produtivo

Trabalhando em Grupo

O Código não é feito apenas para **você** *ler.*
O Código é **para o Grupo**.

Quando escrever código, pense em como facilitar o entendimento do **Grupo**.

Modularização

- propriedades (1)

Modularização

- propriedades (1)
- implementações (métodos) (2)

Modularização

- propriedades (1)
- implementações (métodos) (2)
- encapsulamento de (1)+(2)

Modularização

- propriedades (1)
- implementações (métodos) (2)
- encapsulamento de (1)+(2)
- reutilização

Estrutura de um programa

- preprocessor (♯pragmas, ♯includes, ♯defines)

Estrutura de um programa

- preprocessor (♯pragmas, ♯includes, ♯defines)
- prototypes

Estrutura de um programa

- preprocessor (♯pragmas, ♯includes, ♯defines)
- prototypes
- global vars (horror!)

Estrutura de um programa

- preprocessor (♯pragmas, ♯includes, ♯defines)
- prototypes
- global vars (horror!)
- main

Estrutura de um programa

- preprocessor (♯pragmas, ♯includes, ♯defines)
- prototypes
- global vars (horror!)
- main
- demais funções

C é um Espaço Vetorial

- do while for (laços)

C é um Espaço Vetorial

- do while for (laços)
- break continue return (controle de fluxo)

C é um Espaço Vetorial

- do while for (laços)
- break continue return (controle de fluxo)
- if else (condicionais)

C é um Espaço Vetorial

- do while for (laços)
- break continue return (controle de fluxo)
- if else (condicionais)
- switch case default (multipla escolha)

C é um Espaço Vetorial

- do while for (laços)
- break continue return (controle de fluxo)
- if else (condicionais)
- switch case default (multipla escolha)
- enum struct union (estruturas)

C é um Espaço Vetorial

- do while for (laços)
- break continue return (controle de fluxo)
- if else (condicionais)
- switch case default (multipla escolha)
- enum struct union (estruturas)
- typedef goto (extensões)

Maldição

goto

Lembrem-se: Quem desse comando se utilizar, há de sofrer inenarráveis tormentos no profundo poço da desgraça eterna.

Morfologia de Novo

Em C, cada bloco de código isolável do resto do programa é chamado de **bloco funcional** e deve respeitar duas características básicas que compõem o formalismo visual:

1. delimitadores do bloco alinhados verticalmente
2. bloco respeita hierarquia horizontal

Finalmentes

Roberto Parra nUSP 1694869

esta apresentação foi totalmente elaborada em **LaTeX**, no **slackware** linux ftrn

- 100% Free Software
- 100% MS free !
- 100% elétrons recicláveis

e lembrem-se: Heisenberg *pode* ter estado aqui !