Introdução à Ciência da Computação

Introdução à Linguagem C: Parte I

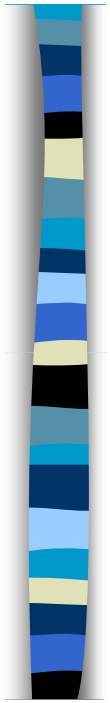
Prof. Ricardo J. G. B. Campello

Créditos

- Alguns slides a seguir foram adaptados dos originais gentilmente cedidos por:
  - Prof. André C. P. L. F. Carvalho

Sumário

- Breve Histórico
- Princípios Básicos
- Tipos de Dados
- Constantes e Variáveis
- Noções de Entrada e Saída



A Origem da Linguagem C

- Linguagem BCPL
  - *Basic Combined Programming Language*
  - Desenvolvida em 1967
  - Foi refinada para uma linguagem chamada B
    - Ken Thompson (Bell Laboratories)
  - Em 1972, Dennis Richie (Bell Labs) melhorou a linguagem B para formar a linguagem C tradicional
    - C foi concebida como a linguagem para o desenvolvimento do sistema operacional Unix
    - Livro “The C Programming Language” de co-autoria de Richie atraiu uma grande atenção à linguagem C

A Origem da Linguagem C

- Muitos compiladores C foram então desenvolvidos para os diferentes tipos de computadores
- A rápida expansão de C levou a um grande número de variações na linguagem original
  - Semelhantes, mas incompatíveis
- Necessidade de uma versão padrão

A Origem da Linguagem C

- Em 1983 foi criado um comitê técnico do *American National Standards Institute* (ANSI)
- Objetivo
  - propor uma definição da linguagem C que fosse não ambígua e independente da arquitetura do computador
- Era criado então o padrão **C ANSI**

Criando um Programa C

- Passos
  - Edição
  - Compilação
    - Pré-processamento
    - Compilação
    - Link-edição
  - Execução

Criando um Programa C

- Edição
  - Usando um editor de textos, escreve o texto do programa em um arquivo
  - Arquivo é chamado de *código fonte*
  - Nome do arquivo em geral termina com “.c”
    - Exemplo: primeiro.c

Meu Primeiro Programa em C

Programa ***primeiro.c***

```
/* primeiro programa que escrevi
na linguagem C */

/* inclusão de biblioteca */
#include<stdio.h>

void main(void) {
    printf("Funcionou!!!");
}
```

Comentários do programa

Diretiva de compilação

Programa principal

Criando um Programa C

- Compilação
  - No sentido estrito, produz um código (conjunto de instruções) em linguagem de máquina a partir do código fonte
    - *Arquivo objeto* (p. ex. primeiro.o ou .obj)
  - No sentido amplo, também liga este código com outros códigos utilizados por ele (link-edição), gerando um programa executável
    - *Arquivo executável* (p. ex. primeiro.exe ou .out)

Criando um Programa C

- Compilação
  - Em alguns ambientes integrados de desenvolvimento (IDEs), basta abrir o arquivo fonte e selecionar a opção de "compilação"
  - Automaticamente o arquivo executável é gerado, incluindo as funções utilizadas de **bibliotecas** da linguagem

Criando um Programa C

- Execução
  - Duplo clique no ícone do programa executável
  - ... ou digitar o nome do programa na linha de comando do sistema operacional
    - Exemplo: > primeiro.exe (ENTER)
  - ... ou selecionar a opção "executar" após compilar o programa em um IDE

Estrutura de um Programa C

- Estrutura básica de um programa em C:

```
Diretivas de Compilação (p. ex. uso de bibliotecas)
Definição de Tipos de Dados;
Definição/Declaração de Constantes/Variáveis Globais;
Declaração (Protótipo) de Funções;

void main(void){
   Declaração de variáveis locais;
   Instruções (comandos);
}

Definição de Funções;
```

Diretivas de Compilação

- Diretivas de compilação não fazem parte do código em si:
  - Apenas estabelecem opções de compilação
    - Como o pré-processador deve preparar o código fonte para que este possa ser compilado para código objeto
    - Exemplos
      - Compilação condicional
      - Inclusão de códigos fonte externos, como, por exemplo, cabeçalhos de bibliotecas (arquivos .h)
      - ...

Delimitação de Instruções

- Instruções em C terminam com ponto-e-vírgula
  - Este caractere (;) serve, portanto, para separar duas instruções distintas no programa
- Retorno de Carro (↵) não é delimitador!
  - Ou seja, não adianta mudar de linha ("Enter") para dizer que se quer iniciar outra instrução

Identificadores

- Um identificador (de constante, variável, função, etc) é formado por:
  - uma única letra; ou
  - uma letra seguida de outras letras e/ou dígitos
- Não são permitidos espaço em branco e caracteres especiais
  - Exceção: *underline* (_) → Exemplo: A_1;
- C ANSI permite identificadores de qualquer tamanho
  - Porém, o no. de caracteres significativos depende do compilador
- Identificadores com maiúsculas e minúsculas são diferentes!
  - ident ≠ IDENT ≠ IDeNt

Palavras Reservadas

- **Palavras Reservadas** (**palavras-chave**) **C ANSI**:

auto	**double**	**int**	**struct**
break	**else**	**long**	**switch**
case	**enum**	**register**	**typedef**
char	**extern**	**return**	**union**
const	**float**	**short**	**unsigned**
continue	**for**	**signed**	**void**
default	**goto**	**sizeof**	**volatile**
do	**if**	**static**	**while**

- Não podem ser usadas como nomes de identificadores

Atribuição

- Atribuição é feita via operador "="
  - Equivale ao ← do pseudo-código
- Exemplo, sejam A, B e C variáveis inteiras:

  A = 5;

  B = 3;

  C = (A + B) * 2 / 4;
- Portanto, as expressões acima são **atribuições** em C, não são expressões relacionais!
  - O operador relacional de igualdade em C é "==" (vide a seguir)

Operadores Básicos

- Operadores aritméticos:
  - –, +, *, /, %, – –, ++
- Operadores relacionais:
  - >, <, >=, <=, ==, !=
- Operadores lógicos:
  - &&
  - ||
  - !

Notas (Operadores Aritméticos)

- % é o operador de módulo (resto de divisão inteira)
- Não existe operador específico para divisão inteira
  - Operador de divisão (/) realiza divisão inteira quando aplicado sobre inteiros
  - Por exemplo, 5 / 2 retorna 2!
  - O mesmo vale para x / y se x e y forem do **tipo** inteiro
- É preciso especificar se isso não é desejado...
  - Exemplo: **float** z; z =(**float**) 5 / 2;

Notas (Operadores Aritméticos)

- Operadores de incremento / decremento:
  - Se x é uma variável do tipo inteiro, então
    - Os comandos x++ ou ++x são equivalentes a x = x + 1;
    - Os comandos x-- ou --x são equivalentes a x = x – 1;
  - Exemplo:

```
#include<stdio.h>
void main(void){
  int x, y, z;
  x = 5;
  y = x++;
  z = ++x;
  printf("%d %d %d", x, y, z);
}
```

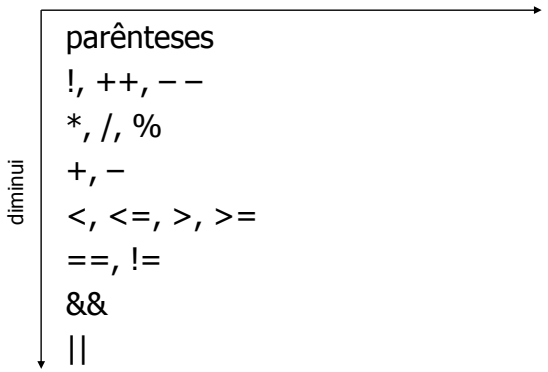
imprime na tela 7 5 7

Notas (Operadores Aritméticos)

- Abreviações:
  - x += 100 é equivalente a x = x + 100
  - x –= 50 é equivalente a x = x – 50
  - x *= 30 é equivalente a x = x * 30
  - x /= 10 é equivalente a x = x / 10

Expressões

- Prioridades dos Operadores Básicos



Funções Pré-Definidas

- Existem diversas, organizadas em bibliotecas
  - Funções matemáticas
  - Funções para manipulação de strings
  - etc.
- Veremos algumas delas em nossos exemplos ao longo do curso, conforme a necessidade

Comentários

- /* ... */
- Exemplo:

  /* Isso é um comentário que começa nessa linha
  e continua nessa outra linha, terminando aqui */
- Não terminam com ponto-e-vírgula
  - pois não são comandos (instruções)

Tipos de Dados

- Tipos Básicos Pré-Definidos (tipos primitivos):
  - Inteiros (**int**) e Reais (**float** ou **double**)
  - Caracteres (**char**)
- Variações dos anteriores através de modificadores
- Tipo especial nulo (**void**)
- Tipos Criados pelo Usuário (aula posterior...):
  - **Variáveis compostas homogêneas**
    - Vetores, matrizes e arranjos multi-dimensionais
  - **Variáveis compostas heterogêneas**
    - Registros e conjuntos de registros

Notas (Tipos de Dados)

- Em princípio, C não possui um tipo lógico pré-definido
- Em expressões lógicas e relacionais:
  - O valor 0 (**zero**) é tratado como **Falso**
  - Qualquer valor **não nulo** (p. ex. 1) é tratado como **Verdadeiro**
- Por exemplo, a expressão relacional (3 < 5) retorna um valor numérico não nulo, pois é verdadeira
- Por outro lado, a expressão lógica (1 && 0) retorna zero, pois Verdadeiro e Falso é Falso

Notas (Tipos de Dados)

- C também não possui um tipo *string* pré-definido
- Strings em C são tratadas como **vetores de caracteres**
- Veremos vetores (e strings) mais adiante no curso...

Caracteres

- Padrão da linguagem é o tipo primitivo **char**
  - Ocupa 1 byte
  - Tecnicamente, C manipula o valor deste byte como um número entre 0 e 255 associado a um dado caractere (tabela ASCII estendida)
  - Logo, as duas atribuições no programa abaixo são equivalentes e o mesmo caractere ("A") é impresso na tela 2 vezes:

```
#include<stdio.h>
void main(void){
  char c1, c2;
  c1 = 'A';
  c2 = 65;
  printf("%c %c", c1, c2);
}
```

Inteiros

- Padrão da linguagem é o tipo primitivo **int**
  - Tamanho em bytes depende da arquitetura
  - Faixa **mínima** (C ANSI): 2 bytes → faixa de –32768 a 32767
  - Serve como referência para modificações
- Principais Modificadores (efeito depende do compilador...):

long int	mesmo tamanho em bytes que int ou o dobro
short int	mesmo tamanho em bytes que int ou a metade
unsigned int	mesmo tamanho em bytes que int (mas sem sinal)

Reais (ponto flutuante)

- Padrão da linguagem são os tipos primitivos
  - **float**: faixa mínima 32 bits (4 bytes)
  - **double**: faixa mínima 64 bits (8 bytes)
- Principal extensão via modificador:
  - **long** double (maior precisão possível)

Sintaxe de Inteiros e Reais

- Sintaxe de Valores Inteiros em C:
  - Sequência de dígitos 0 a 9 precedida ou não de sinal + ou –
  - Exemplos: 10, +10, –3, 0, 30000
- Sintaxe de Valores Reais em C:
  - Sequência de dígitos 0 a 9 precedida ou não de sinal + ou – , seguida ou não de ponto decimal e outra sequência de dígitos
  - Exemplos: 10.0, –10.5, 3.141592
- Podem ainda terminar com a letra E seguida de outra sequência de dígitos precedida ou não de sinal + ou –
  - Exemplos: 7E5, 21.0E3, 21E+3, 21E–3, 7.4567E2

Sintaxe de Inteiros e Reais

- Exemplo:

```
#include<stdio.h>
void main(void){
  int x;
  float y;
  x = 5E2;
  y = 5.6E-2;
  printf("%d %f", x, y);
}
```

imprime na tela 500 0.056

Notas (Operadores Aritméticos)

- Conversões Automáticas de Tipo por Promoção:
  - Antes de realizar uma operação aritmética, C converte internamente ambos os operandos no maior tipo
  - Assim, os dois operandos passam a possuir o mesmo tipo, que será também o tipo do resultado
    - É por esta razão que a divisão de dois inteiros é inteira, exceto se uma conversão forçada de tipo (**cast**) for realizada
  - Em expressões, isso é realizado operação por operação

Constantes

- Definição:
  - identificador associado a um valor inicial que não pode ser modificado pelo programa
- Declaração de Constantes:
  - **const** tipo identificador = valor;
- Exemplo:
    - **const float** pi = 3.141592;
- Principais Utilidades:
  - Clareza de código
  - Prevenção de erros

Variáveis

- Declaração de Variáveis
  - **tipo** identificador(es);
- Exemplos
  - **char** d;
  - **int** idade;
  - **float** b1, B1, salario;
  - **double** A_f;
- É possível inicializar variáveis ao declarar
  - Exemplos: **char** d = 'a'; **int** idade = 30;

Entrada e Saída

- Existem diversas funções pré-definidas para entrada e saída de dados em linguagem C
- Por ora, precisamos de apenas duas dessas funções para implementar nossos programas
  - **printf** e **scanf**
- Ao longo do curso poderemos utilizar outras, caso seja necessário ou mais apropriado

Saída de Dados com `printf`

- **printf** é uma função bastante flexível que permite a escrita de dados formatados de diferentes modos
- Veremos a seguir as formas mais elementares
  - Escrita de Texto (constante string)
    - Texto a ser exibido vai entre aspas
      - Exemplo: `printf("Aqui vai o texto!");`
    - Admite **caracteres especiais**, como quebra de linha (**\n**)
      - Exemplo: `printf("Primeira linha\nProxima linha");`

Saída de Dados com `printf`

- Formas mais elementares de `printf`
  - Escrita de Caracteres (**char**)
    - Utiliza-se o código de formato **%c**
    - Exemplo:

```
#include<stdio.h>
void main(void){
  char x;
  x = 'S';
  printf("Fumante = %c", x);
}
```

Saída de Dados com `printf`

- Formas mais elementares de `printf`
  - Escrita de Inteiros (**int**)
    - Utilizam-se os códigos de formato **%d** ou **%i** (equivalentes)
      - Nota: %hd ou %hi para **short int** e %ld ou %li para **long int**
    - Exemplo:

```
#include<stdio.h>
void main(void){
  int id = 40, no_filhos = 2;
  printf("Idade = %d\nNumero de Filhos = %d", id, no_filhos);
}
```

Saída de Dados com `printf`

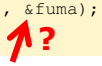
- Formas mais elementares de `printf`
  - Escrita de Reais (**float** e **double**)
    - Utiliza-se o código de formato **%f**
      - Nota: %Lf para **long double**
    - Exemplo:

```
#include<stdio.h>
void main(void){
  double sal = 3435.26;
  printf("Salario = %f", sal);
}
```

Entrada de Dados com `scanf`

- **scanf** é uma função bastante flexível que permite a entrada de dados formatados de vários modos
- Veremos a seguir as formas mais elementares
  - Leitura de Caractere*
    - Utiliza-se o código de formato **%c**
      - Exemplo:

```
#include<stdio.h>
void main(void){
  char fuma;
  printf("Fumante? (S / N):");
  scanf("%c", &fuma);
}
```



* PS: `scanf` pode não ser a função mais apropriada para ler caracteres, especialmente em ambientes iterativos.

Entrada de Dados com `scanf`

- Formas mais elementares de `scanf`
  - Leitura de Inteiros (**int**)
    - Utilizam-se os códigos de formato **%d** ou **%i** (equivalentes)
      - %hd e %ld (ou %hi e %li) para **short int** e **long int**, respectivamente
    - Exemplo:

```
#include<stdio.h>
void main(void){
  int id;
  printf("Entre com a idade (e tecle ENTER): ");
  scanf("%d", &id);
  printf("\nA idade informada = %d", id);
}
```

Entrada de Dados com `scanf`

- Formas mais elementares de `scanf`
  - Leitura de Reais (**float**)
    - Usam-se os códigos de formato **%f**, **%e** ou **%g** (equivalentes)
    - Exemplo:

```
#include<stdio.h>
void main(void){
  float val;
  printf("Entre com o valor (e tecle ENTER): ");
  scanf("%f", &val);
  printf("\nO valor informado = %f", val);
}
```

Entrada de Dados com `scanf`

- Formas mais elementares de `scanf`
  - Leitura de Reais de Dupla Precisão (**double**)
    - Usa-se o código de formato **%lf** (**%LF** para **long double**)
    - Exemplo:

```
#include<stdio.h>
void main(void){
  double val;
  printf("Entre com o valor (e tecle ENTER): ");
  scanf("%lf", &val);
  printf("\nO valor informado = %f", val);
}
```

Exercício 1

- Faça um algoritmo em pseudo-código que leia dois valores reais e escreva a raiz cúbica da soma deles
- Transcreva o algoritmo acima para C
  - Dica: Você pode incluir a biblioteca de funções matemáticas da linguagem no início do seu programa, via `#include<math.h>`, e utilizar a função pré-definida `pow(b,e)`, que recebe uma base `b` e um expoente `e` (valores reais do tipo **double**) e calcula b^e

Exercício 2

- Faça um algoritmo que primeiro pergunte ao usuário os seguintes dados: altura, peso, idade e se é fumante ou não (S ou N). Tais dados devem ser armazenados em variáveis correspondentes. Em seguida o algoritmo deve escrever os dados na tela
- Transcreva o algoritmo para C

Exercícios Adicionais

- Traduza para C os pseudo-códigos dos exemplos de estrutura sequencial (aula de estruturas de controle)
  - Compile e execute os programas em laboratório ou no seu computador pessoal para se certificar que funcionam!
  - Pesquise as bibliotecas C caso alguma função seja necessária (por exemplo, raiz quadrada – vide função `sqrt` na biblioteca de funções matemáticas math.h)