LINGUAGEM DE PROGRAMAÇÃO II

Programa

- Sintaxe
- Funções e Recursividade
- Vetores e Matrizes
- Ponteiros
- Registros
- Arquivos Texto
- Arquivos Binário
- Bibliotecas de funções

Avaliações

- Duas provas teóricas (P1 e P2)
- Listas de exercícios (T)
- Média = (2*P1 + 2*P2 + T) / 5

Avaliações

- Datas das avaliações:
  - 29/03: P1
  - 30/04: P2
  - 05/05: Optativa

- As listas de exercícios são entregues a cada semana e não serão aceitas fora do prazo.

Dinâmica

- Enquanto as aulas forem remotas, seguiremos o planejamento:
  - Terças-feiras => síncrono (aulas expositivas)
  - Quintas-feiras => assíncrono (resolução e envio das listas de exercícios)

Programa

- Sintaxe
- Funções e Recursividade
- Vetores e Matrizes
- Ponteiros
- Registros

P1

- Arquivos Texto
- Arquivos Binário
- Bibliotecas de funções

P2

Bibliografia

- DEITEL, H. M.; DEITEL, P. J, “Como programar em C”, 6ª Edição, Pearson, 2011

- SCHILDT, H, C – Completo e Total, Makron Books, 3a Edição, Pearson,1996.

- Apostila “ Linguagem C” – Silvio do Lago Pereira

Contatos

- Profª. Gizelle

- E-mail: kupac@ufrrj.br

LINGUAGEM C

Aula 1

A Linguagem C

- Em 1972, Dennis Richie lançou a linguagem C, a partir da linguagem B.
- No início, C ficou conhecida como a linguagem de desenvolvimento do Unix.
- A expansão do C para hardwares diferentes levou a um grande número de variações na linguagem original.
- Na década de 1980 foi criada a padronização ANSI C.

Vantagens da Linguagem C

- Eficiência:
  - produz programas que consomem pouca memória e rodam rápido;
- Portabilidade:
  - O mesmo programa C pode ser executado em máquinas e sistemas operacionais diferentes;
- Compacta:
  - Vários softwares básicos, sistemas de tempo real e programas de computação gráfica são escritos em C e seus derivados;
  - É uma linguagem não-prolixa.
- Velocidade:
  - Na área de software básico, a velocidade de execução é crítica;

Vantagens da Linguagem C

- Conjunto de operadores de C é muito poderoso;
- Várias operações podem ser combinadas em um único comando;
- Gera códigos menores e mais elegantes;
- Aumenta a produtividade dos programadores;

Vantagens da Linguagem C

- Modularidade:
  - C incentiva a decomposição de um programa em módulos;
  - Diferentes partes do código podem ser escritos por programadores diferentes;
  - Torna a manutenção do código mais simples
  - Facilita a reutilização de código;

- Códigos reutilizados correspondem a mais de 90% dos novos sistemas.

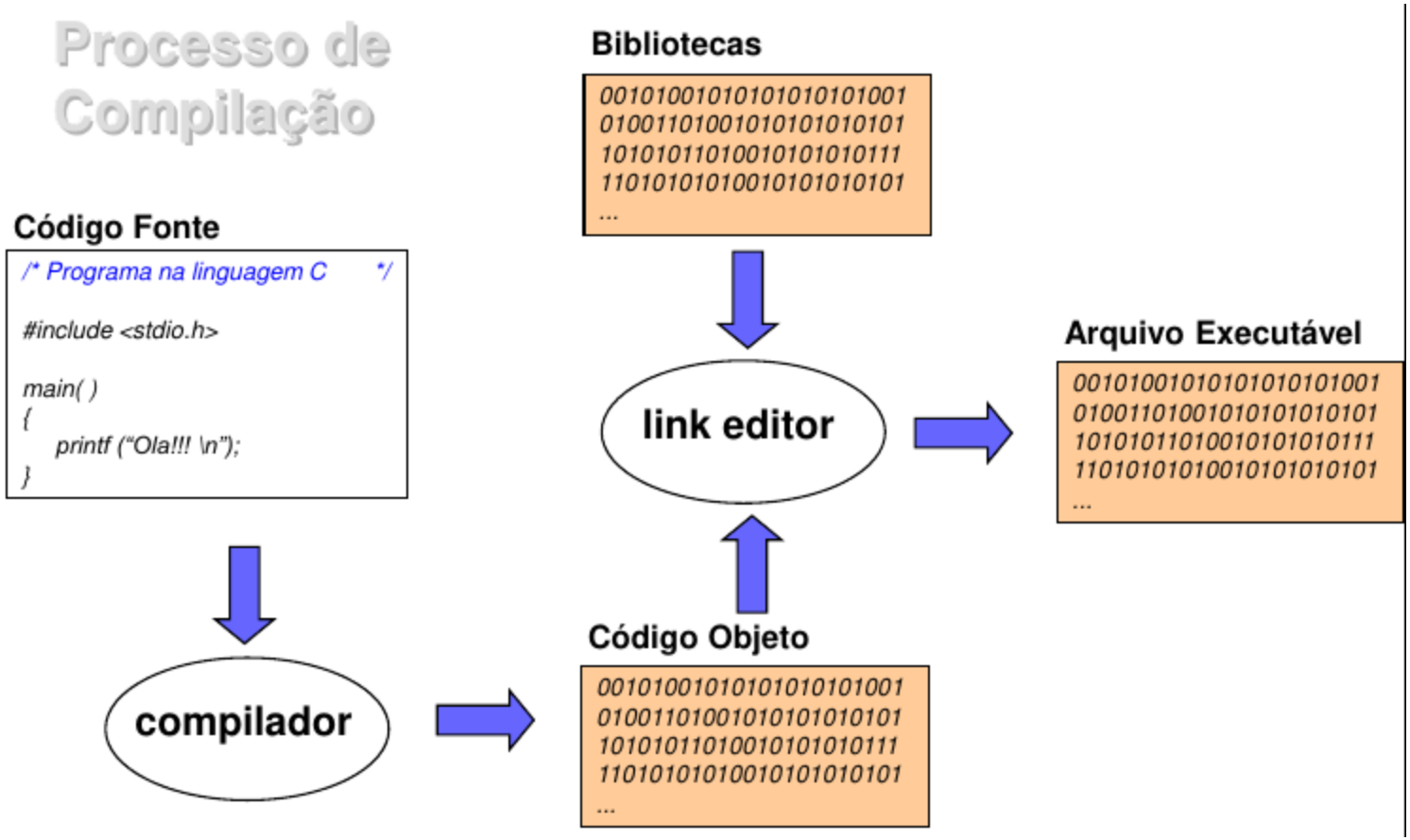
Desvantagens da Linguagem C

- Sintaxe complexa (se comparada ao Pascal);
- Sua flexibilidade pode levar o programador a erros;
- Por não ser uma “linguagem da moda” existem poucas bibliotecas públicas disponíveis e o programador precisa ser “roots” para resolver um problema.

O arquivo .EXE

- Após a escrita do programa em código fonte, alguns passos se seguem para a criação de um arquivo executável:
  - Compilação
  - Link-edição
  - Geração de código binário (executável)

O arquivo .EXE



Estrutura de um programa em C

cabeçalho

```
main ()
{
        <cmd>;
        <cmd>;
        <cmd>;
}
```

Exemplo:

```
#include <stdio.h>

main ()
{
    printf (“Bem vindo ao C!\n”);
}
```

Mais exemplos

```
/* Segundo programa em C */
#include <stdio.h>

main()
{
   printf (“Bem vindo”);
   printf (“ao C!\n”);
}
```

```
 /* Terceiro programa em C */
#include <stdio.h>

main()
{
   printf (“Bem vindo \nao \nC!\n”)
}
```

SINTAXE

Mais exemplos

```
/* Programa  ex_1.c      */

#include <stdio.h>

main ( ) {
     int n, i, soma;
     float final;

     n = 4;
    soma = 0;
     for (i = 1; i < n; i++){
          soma = soma + i;
     }
     final = soma/(n-1);
     printf ("Resultado = %f", final);
}
```

Diretiva *#include*

- A diretiva *#include* permite incluir uma biblioteca.
- Bibliotecas contêm funções pré-definidas, utilizadas nos programas.
- Exemplos:

`#include <stdio.h>`	Funções de entrada e saída
`#include <stdlib.h>`	Funções padrão
`#include <math.h>`	Funções matemáticas
`#include <system.h>`	Funções do sistema
`#include <string.h>`	Funções de texto

Primeiros Conceitos

- Nos programas escritos em C, o módulo principal deverá estar dentro de uma função especial chamada ***main***. Todos os programas em C começam a sua execução a partir da **main()**.

- Os programas em C permitem que voa inclusão de linhas de comentários dentro do código fonte, usando // ou /* */

- ***Preste atenção:***
  - é preciso terminar cada linha do programa com o sinal de **;**
  - O C diferencia as letras maiúsculas das minúsculas. Logo, as variáveis de nome itema e itemA são diferentes!

Exemplo

```
/* Primeiro programa em C */
#include <stdio.h>

main()
{
        printf (“Bem vindo ao C!\n”);
        exit 0;
}
```

- Repare na sequência \n
- A barra invertida indica uma sequência de escape, que é um comando enviado para a saída do programa.
- Por exemplo, \n significa que o cursor irá pular uma linha após escrever a mensagem.

Sequencias de Escape

Sequência	Significado
\n	Nova linha
\t	Tabulação horizontal
\r	Carriage return (CR). Coloca o cursor no início da linha atual (não pula a linha)
\a	Faz soar a campainha do sistema
\\	Imprime um caracter de barra invertida
\”	Imprime um caracter de aspas duplas

Sintaxe Básica

- Comentários em bloco são delimitados por **/*... */**
- Comentários de uma linha são iniciados por **//**
- Atribuição é feita com o operador **=**
- A comparação é feita com o operador **==**
- Exemplo, sejam A, B e C variáveis inteiras:
  - B = 3;
  - A = 5 ;
  - C = (A + B) * 2 / 4;
  - A == B (falso)

Identificadores

- Um identificador é formado por:
  - uma única letra; ou
  - uma letra seguida de outras letras e/ou dígitos;

- Não são permitidos espaço em branco nem caracteres especiais;
  - Exceção: underline (_) → Exemplo: A_1;

- A linguagem C padrão (ANSI C) permite identificadores de qualquer tamanho. Porém, o número de caracteres significativos depende do compilador;

Tipos de Dados

- Tipos primitivos:
  - Inteiros (**int**) e Reais (**float ou double**)
  - Caracteres (**char**)
- Tipo especial nulo (**void**);
- Tipos criados pelo usuário:
  - Variáveis compostas homogêneas
    - Vetores, matrizes e arranjos multi-dimensionais
  - Variáveis compostas heterogêneas
    - Registros e conjuntos de registros

Tipos primitivos modificados

- São variações dos tipos primitivos, realizadas através de modificadores.

- Principais modificadores:
  - 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)

- OBS: o efeito da aplicação de um modificador irá depender do compilador

Tipos Lógicos

- C não possui um tipo lógico pré-definido (true ou false);
- 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 V e F => F

Números reais

- Também conhecidos como números de ponto flutuante
  - float: 4 bytes
  - double: 8 bytes

- Principal modificador:
  - long double maior precisão possível, depende do compilador

Strings

- C também não possui um tipo string pré-definido, o que não significa que não existam funções para trabalhar com elas.

- Strings em C são definidas como vetores de caracteres.

Declaração de Variáveis

- Declaração de Variáveis:
  - tipo identificador(es);

- Exemplos
  - char d;
  - int idade;
  - float b1, B1, salario;
  - double A_f;
  - char d = 'a';
  - int idade = 30;

É possível inicializar as variáveis durante a inicialização!

Constantes

- Definição:
  - identificador associado a um valor inicial que não pode ser modificado pelo programa;
- Declaração:
  - const tipo identificador = valor;
- Exemplo:
  - const float pi = 3.141592;

Funções de Entrada e Saída

- A função scanf():
  - permite que um valor seja lido do teclado e armazenado numa variável;
- Sintaxe:
  - scanf(“formatação”, arg1 , arg2 , ..., argn);
- Exemplo:
  - int idade;
  - char pcd;
  - ...
  - scanf(“%d %c”, &idade, &pcd);

> *O operador & será explicado quando estudarmos ponteiros. Por hora, basta saber que indica a posição de memória de uma variável.*

Funções de Entrada e Saída

- A função printf( ):
  - permite exibir informações formatadas no vídeo;
- Sintaxe:
  - printf(“formatação”, arg1 , arg2 , ..., argn);
- Exemplo:
  - int idade;
  - char pcd;
  - printf(“%d %c”, idade, pcd);

Funções de Entrada e Saída

Printf

```
Printf (“texto”) ;
Printf (“texto\n”) ;
Printf (“texto\t”) ;
Printf (“t\ne\nx\nt\no\n”) ;
Printf (“O número %d é maior do que %d”, varA, varB);
```

Scanf

```
Scanf (“%d”, &varA);
Scanf (“%d %d”, &varA, &varB);
Scanf (“%d%d”, &varA, &varB);

Scanf (%f”, &varC);
Scanf (%s”, &varD);
```

Caracteres de Formatação de Saída

Especificador	Representa
%c	um único caracter
%o, %d, %x	um número inteiro em octal, decimal ou hexadecimal
%u	um número inteiro em base decimal sem sinal
%ld	um número inteiro longo em base decimal
%f, %lf	um número real de precisão simples ou dupla
%s	uma cadeia de caracteres (*string*)
%%	um único sinal de porcentagem

Operadores Aritméticos

Operação	Operador Aritmético	Expressão em C
Adição	+	3 + a
Subtração	-	2 – c
Multiplicação	*	A * D
Divisão	/	X / Y
Resto (Módulo)	%	10 % 2 a % 3

Operadores Binários

- As linhas:

```
total = total + grau;
contador = contador + 1;
```

- podem ser substituídas por:

```
total += grau;
contador += 1;
```

- De fato, essa compactação pode ser feita com qualquer operador binário, conforme descrito na tabela abaixo:

Operador	Exemplo	Formato por extenso
+=	c += 7	c = c + 7
-=	d -= 4	d = d – 4
*=	e *= 5	e = e * 5
/=	f / = 3	f = f / 3
%=	g %= 2	g = g % 2

Operadores de Incremento e Decremento

- Existe ainda outra forma de compactar uma expressão, utilizada quando o incremento ou decremento é de apenas uma unidade:

Operador	Exemplo	Comportamento
++	++a	Incrementa a de 1 e depois usa o novo valor na expressão onde a se localiza.
++	a++	Usa o valor de a na expressão onde a se localiza e depois incrementa de 1.
--	--a	Decrementa a de 1 e depois usa o novo valor na expressão onde a se localiza.
--	a--	Usa o valor de a na expressão onde a se localiza e depois decrementa de 1.

Exemplo 1

```
#include <stdio.h>

main () {
        int c;

        c = 5;
        printf("%d\n", c);

        /* pós-incremento */
        printf("%d\n", c++);
        printf("%d\n\n", c);

        c = 5;
        printf("%d\n", c);

        /* pré-incremento */
        printf("%d\n", ++c);
        printf("%d\n\n", c);

}
```

O que será impresso:

5
5
6

5
6
6

Exemplo 2

```
#include<stdio.h>

void main(void){
    int x, y, z;

    x = 5;
    y = x++;
    z = ++x;
    printf("%d %d %d", x, y, z);
}
```

O que será impresso:

7

5

7

Operadores Relacionais

Operação	Operador em C	Exemplo
Igualdade	= =	x = = y
Desigualdade	!=	x != y
Maior que	>	x > y
Menor que	<	x < y
Maior ou igual	>=	x >= y
Menor ou igual	<=	x <= y

Operadores Lógicos

Operador	Função	Exemplo
&&	E	if (a==1 && b == 3)
||	OU	if (a==1 || b == 3)
= =	IGUALDADE	if (a==1)
!	NEGAÇÃO	if (!a && b == 3)

- Observação:
  - No C, qualquer valor diferente de 0 é considerado VERDADEIRO.
  - Então, um valor ***!a*** será igual a:
    - FALSO, se ***a*** for VERDADEIRO
    - VERDADEIRO, se ***a*** for FALSO.