

Instituto Federal Sudeste de Minas Gerais

Campus Barbacena

Curso de Tecnologia em Sistemas para Internet

Disciplina:Lógica de Programação

Prof. Wender Magno Cota

Assunto: Introdução a Linguagem C

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

Criação de um programa em C

Passos

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

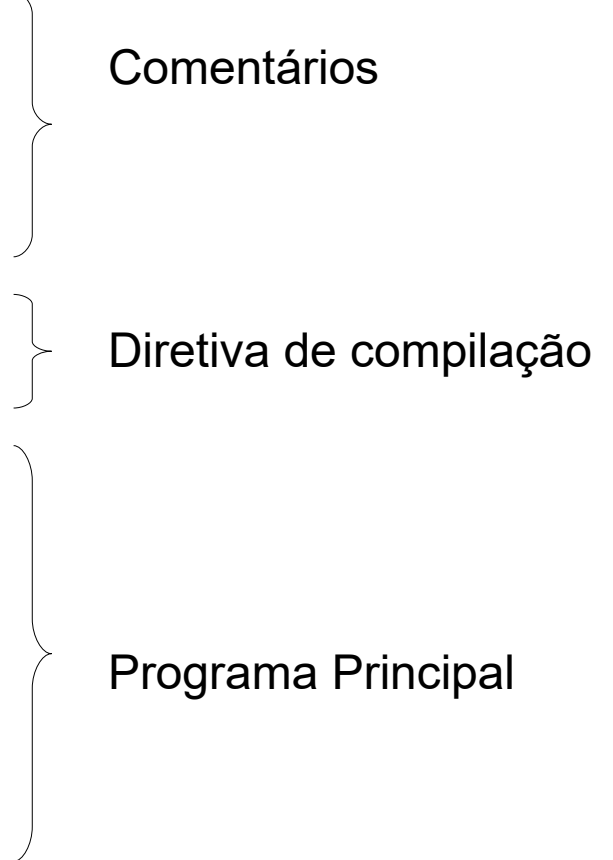
Criação de um Programa em C

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

Primeiro programa em C

```
/*Primeiro programa escrito por mim  
usando a Linguagem de Programação C*/  
/* Inclusão de Biblioteca */  
#include <stdio.h>  
int main()  
{  
    printf("Olá Mundo !!!");  
    return 0;  
}
```



Comentários

Diretiva de compilação

Programa Principal

Criação de um programa em 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)

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

Identificador

Sequência de letras, números e underscore(underline) usada para nomear variáveis, funções, etc.

Regras:

- a) Deve começar com uma letra ou _ (sublinhado ou underscore);
- b) Os demais símbolos, se existirem, podem ser letras, números ou underscore;
- c) Não é permitido o uso de símbolos especiais, espaço em branco e nem acentuação;
- d) Não é permitido o uso de palavras reservadas.

Palavras Reservadas

auto break case char

const continue default do

double else enum extern

float for goto if

int long register return

short signed sizeof static

struct switch typedef union

unsigned void volatile while

Variáveis

- Variáveis em um programa C estão associadas a posições de memória que armazenam informações
- Nomes de variáveis tem que ser um identificador válido
- Em C, apenas os 31 primeiros caracteres são considerados

Variáveis

Exemplos de nomes de variáveis:

Corretos

Contador

Teste23

Barbacena_Querida

__sizeint

Incorretos

1contador

oi!gente

Barbacena..Querida

__size-int

- Palavras-chave (reservadas) de C não podem ser utilizadas como nome de variáveis: int, for, while, etc...
- C é case-sensitive:

contador \neq Contador \neq CONTADOR

Tipos de Dados Básicos

- char: um byte que armazena o código de um caracter do conjunto de caracteres local
- int: um inteiro cujo tamanho depende do processador, tipicamente 16 ou 32 bits
- float: um número real com precisão simples
- double: um número real com precisão dupla
- void: vazio(inglês)

Modificadores de Tipos

- Modificadores alteram algumas características dos tipos básicos para adequá-los a necessidades específicas
- Modificadores:
 - signed: indica número com sinal (inteiros e caracteres)
 - unsigned: número apenas positivo (inteiros e caracteres)
 - long: aumenta abrangência (inteiros e reais)
 - short: reduz a abrangência (inteiros)

Alguns Tipos Primitivos

ABRANGÊNCIA DE DADOS: 32 BITS

Tipo	Tamanho(bytes)	Abrangência
char	1	-128 a 127
unsigned char	1	0 a 255
int	4	-2.147.483.648 a 2.147.483.647
unsigned int	4	0 a 4.294.967.295
short int	2	-32.768 a 32.767
long int	4	-2.147.483.648 a 2.147.483.647
unsigned long	4	0 a 4.294.967.295
float	4	3,4E-38 a 3,4E38
double	8	1,7E-308 a 1,7-308
long double	10	3,4E-4932 a 3,4E-4932

Constantes

Constantes são valores fixos que não podem ser modificados pelo programa:

Tipo	Exemplos
char	'a' '\0' '1' 't' '\t'
int	-20 0 189
long int	-20L 1L 189L
float	-1.2F 0.9F 1234.23F
double	1.2 0.9 1234.23
String	"" "Casa" "Galo The Best Of The Best"

Constantes Hexadecimais e Octais

As constantes hexadecimais começam com 0x ou 0X e as octais com 0.

Exemplos:

Constante	Tipo
0xEF	Constante Hexadecimal
0x12A4	Constante Hexadecimal
03212	Constante Octal
034215432	Constante Octal

Constantes Barra Invertida

Usados para representar caracteres que não têm uma representação gráfica. Alguns exemplos:

Código	Significado
<code>\n</code>	Nova linha
<code>\t</code>	Tabulação horizontal
<code>\v</code>	Tabulação vertical
<code>\0</code>	Nulo
<code>\”</code>	Aspas
<code>\'</code>	Apóstrofo
<code>\\</code>	Barra invertida

Declaração de Variáveis

Sintaxe:

tipo lista_variaveis;

onde lista_variáveis pode ser uma única variável ou um conjunto de variáveis separadas por vírgula.

Exemplo:

```
int idade, num_dep;
```

```
float salario;
```

```
char sexo;
```

Saída de Dados

printf

Sintaxe:

```
printf(string_controle, lista_argumentos);
```

string_controle representa o que será impresso além de caracteres de controle indicando onde os valores dos argumentos, caso tenha, serão apresentados.

Saída de Dados

Exemplo:

```
#include <stdio.h>
int main(){
    int i;
    printf("Exemplo de uso do printf \n");
    i=10;
    printf("O dobro de %d = %d",i,i+i);
    return 0;
}
```

Saída:

Exemplo de uso do printf

O dobro de 10 = 20

Saída de Dados

Código	Significado
<code>%c</code>	Um único caractere
<code>%d</code>	Um decimal inteiro
<code>%ld</code>	Um long long(inteiro de 8 bytes)
<code>%f</code>	Um número em ponto flutuante (precisão simples)
<code>%s</code>	Uma string
<code>%lf</code>	Um double
<code>%Lf</code>	Um long double

Entrada de Dados

`scanf("expressão de controle", lista de argumentos);`

Expressão de controle: caracteres lidos do teclado + cód. de formatação dos argumentos

Lista de argumentos: endereços das variáveis a serem lidas

Operador de Endereço &:

Exemplos:

`scanf("%d %d",&A, &B); /*se A e B são int*/`

`scanf("%f %f",&A, &B); /*se A e B são float */`

Entrada de Dados

Os especificadores de formato de entrada são precedidos por um sinal % e dizem à função scanf() qual tipo de dado deve ser lido em seguida. Esses códigos são listados na tabela a seguir.

Código	Significado
%c	Lê um único caractere
%d	Lê um decimal inteiro
%ld	Lê um inteiro(long long)
%f	Lê um número em ponto flutuante com ponto opcional
%s	Lê uma string
%lf	Lê um double
%Lf	Lê um long double
%li ou %Li	Lê um long int

Operadores - Relacionais

Operador	Descrição
>	Maior que
>=	Maior ou igual
<	Menor que
<=	Menor ou igual
!=	Diferente
==	Igual

Operadores - Relacionais

Operadores Lógicos:

Operador	Significado
----------	-------------

&&	E
----	---

	Ou
--	----

!	Não
---	-----

Na linguagem C 0(zero) e nulo são falsos e qualquer outro valor é considerado verdadeiro.

Atribuição

Sintaxe

`variavel = valor;`

Onde valor pode ser uma constante, variável ou expressão. No caso de ser uma expressão, ela será primeiro resolvida e seu resultado armazenado em variável.

Exemplos

```
int main(){
    int a ,b,c;
    a=10; //armazena o valor 10 na variável a
    b =5; //armazena o valor 5 na variável b
    c =b; //é armazenado na variável c o valor de b(5)
    b = b+a+c; //é armazenado na variável b a soma dos valores armazenados nas variáveis b,c e a
    printf("\nValor de a = %d",a);
    printf("\nValor de b = %d",b);
    printf("\nValor de c = %d",c);
    return 0;
}
```

Saída:

Valor de a = 10

Valor de b = 20

Valor de c = 5

Atribuição - Resumida

variavel op=valor;

Exemplo:

$n += 10;$  $n = n + 10;$

Pode usar com qualquer operador aritmético.

Operadores

Operadores são elementos funcionais que atuam sobre operandos e produzem um determinado resultado.

Por exemplo, a expressão $3 + 2$ relaciona dois operandos (os números 3 e 2) por meio do operador (+) que representa a operação de adição.

Operadores

De acordo com o número de operandos sobre os quais os operadores atuam, os operandos podem ser classificados em:

- **Binários**: quando atuam sobre dois operandos. Ex.: os operadores das operações aritméticas básicas (soma, subtração, multiplicação e divisão).
- **Unários**: quando atuam sobre um único operando. Ex.: o sinal de (-) na frente de um número, cuja função é inverter seu sinal.

Operadores

Outra classificação dos operadores é feita considerando-se o tipo de dado de seus operandos e do valor resultante de sua avaliação.

Segundo esta classificação, os operadores dividem-se em aritméticos, lógicos e relacionais. Esta divisão está diretamente relacionada com o tipo de expressão onde aparecem os operadores.

Operadores Aritméticos

Operador	Descrição
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão(inteira)

Obs:

- a) O resultado de uma expressão formada pelo operador / vai depender do tipo dos operandos. Caso os dois sejam inteiros o resultado será a divisão inteira. Neste caso para obter a divisão real deve-se realizar um cast.
- b) Os operandos usados com o operador % devem ser inteiros.

Operadores Aritméticos

Hierarquia dos operadores aritméticos

parênteses
↓
* / %
↓
+ -

Exemplo:

$$2 + 3 * 4 = 14$$

$$(2+3)*4 = 20$$

Ob.: uma expressão formada por operadores com a mesma prioridade será resolvida da esquerda para a direita.

$$20/3*4 = 24$$

$$20/(3*4) = 1$$

Operadores na Linguagem C

- Uma expressão formada por operadores com a mesma prioridade será resolvida da esquerda para a direita;
- Toda variável usada em uma expressão deve ser inicializada (com o comando de atribuição ou com o comando de leitura);
- Linguagem C é case sensitive, ou seja, será feita distinção entre letras maiúsculas e minúsculas (relacionadas a formação de identificadores).

Operadores Relacionais

Os operadores relacionais permitem comparar valores (ou expressões) de tipos compatíveis, devolvendo (gerando) como resultado um valor lógico(verdadeiro ou falso).

Obs.: A Linguagem C não possui o tipo boolean(lógico). Assim sendo, 0(zero é falso e qualquer outro valor é considerado verdadeiro).

O resultado de uma expressão formada por um operador relacional resultará em um valor verdadeiro ou falso.

Operadores Relacionais

Operador	Descrição
==	Igual
!=	Diferente
>	Maior que
>=	Maior ou igual
<	Menor que
<=	Menor ou igual

Nota

Um erro comum em iniciantes na Linguagem C é confundir o uso de = com ==.

O = é atribuição.

Operadores Lógicos

Servem(funcionam) como conectivos para a formação de novas proposições. Iremos trabalhar com três operadores lógicos:

Operador	Significado em Português	Descrição
&&	E	Conjunção
	Ou	Disjunção(não exclusiva)
!	Nao	Negação

Operadores Lógicos

Operador && (E)

É um operador binário. A expressão será verdadeira somente se todos os operandos tiverem o valor verdadeiro.

Tabela verdade do operador &&

A	B	A && B
V	V	V
V	F	F
F	V	F
F	F	F

Operadores Lógicos

Operador II(OU)

É um operador binário. A expressão será verdadeira pelo menos um dos operandos tiver o valor verdadeiro.

Tabela verdade do operador ||

A	B	A B
V	V	V
V	F	V
F	V	V
F	F	F

Portugol – Lógicos

Operador !(NÃO)

É um operador Unário. Nega uma afirmação, invertendo seu valor: se for **verdadeiro** torna-se **falso**, e se for **falso** torna-se **verdadeiro**.

Tabela verdade do operador !

A	! A
V	F
F	V

Operadores - Precedência

() [] -> .
++ -- ! & * ~ . (type) sizeof (operadores unários)
* / %
+ -
<< >>
< <= >= >
== !=
&
&
|
&&
||
?:
= op=
,

Estrutura de Seleção

Comando if

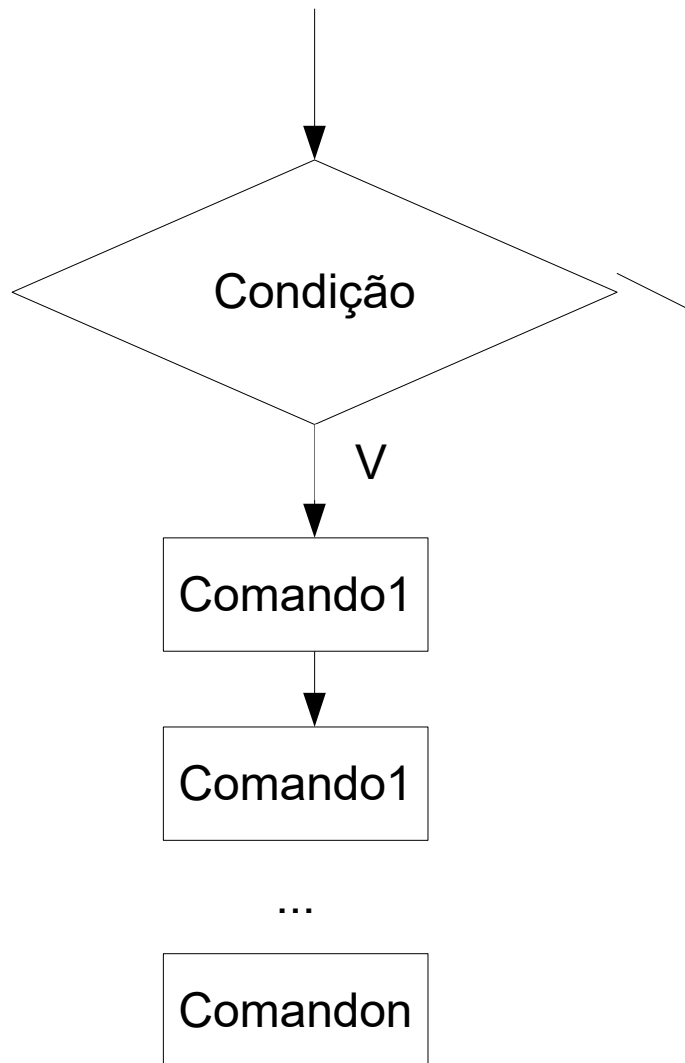
Sintaxe:

```
if(<condição>)  
{  
    ComandoIF1;  
    ComandoIF2;  
    ...  
    ComandoIFn;  
}
```

ou

```
if(<condição>)  
{  
    ComandoIF1;  
    ComandoIF2;  
    ...  
    ComandoIFn;  
}  
else  
{  
    ComandoElse1;  
    ComandoElse2;  
    ...  
    ComandoElsen;  
}
```

Estrutura de Seleção



Estrutura de Seleção - Exemplo

```
#include <stdio.h>
#include <math.h>
int main(){
    float a,b,c,delta,x1,x2;
    printf("Forneça os coeficientes da equação:");
    scanf("%f%f%f",&a,&b,&c);
    if(a==0)
        printf("Não é uma equação do segundo grau");
    else{
        delta = b*b-4*a*c;
        if(delta<0)
            printf("A equação não possui solução real");
        else{
            x1=(-b+pow(delta,0.5))/(2*a);
            x2=(-b-pow(delta,0.5))/(2*a);
            printf("Raízes da equação %.2f e %.2f",x1,x2);
        }
    }
    return 0;
}
```

Estrutura de Seleção

Comando switch: Usado para testar uma expressão em relação a um conjunto de valores pré-estabelecidos.

Sintaxe:

```
switch(<expressão>)  
{  
    case <valor1>: comandos1;  
                break;  
    case <valor2>: comandos2;  
                break;  
    case <valorn>: comandosn;  
                break;  
    default: comandos_default;  
}
```

Estrutura de Seleção - Switch

- A cláusula default é opcional
- O break faz com que a execução do switch seja interrompida.
- Expressão deve ser de um tipo ordinal.

Estrutura de Seleção - Switch

```
#include <stdio.h>
int main(){
    float n1,n2;
    char op;
    printf("Forneça dois números:");
    scanf("%f%f",&n1,&n2);
    printf("Forneça um Operador + - * /:");
    scanf("%c",&op);
    switch (op)
    {
        case '+': printf("Resultado = %f",n1+n2); break;
        case '-': printf("Resultado = %f",n1-n2); break;
        case '*': printf("Resultado = %f",n1*n2); break;
        case '/': if(n2!=0)
                    printf("Resultado = %f",n1/n2);
                else
                    printf("Divisão por zero"); break;
        default: printf("Operador Incorreto");
    }

    return 0;
}
```


Estrutura de Seleção...

Operador Ternário ?

Usado em expressões do tipo

```
if(condição)
```

```
    comando1;
```

```
else
```

```
    comando2;
```

Pode ser escrita como

```
condicao ? comando1 : comando2;
```

Estrutura de Seleção...

Operador Ternário ? - Exemplo 01

```
#include <stdio.h>
```

```
int main(){
```

```
    int n1,n2;
```

```
    printf("Forneça dois números inteiros:");
```

```
    scanf("%d%d",&n1,&n2);
```

```
    (n1>n2) ? printf("%d",n1) : printf("%d",n2);
```

```
    return 0;
```

```
}
```

Algumas Funções Matemáticas – math.h

double pow(double base, double expoente)
Calcula base elevada a expoente: $\text{base}^{\text{expoente}}$

double sqrt(double valor)
Calcula a raiz quadrada de **valor**

double cbrt(double valor)
Calcula a raiz cúbica de **valor**

double exp(double x)
Calcula e^x

double trunc(double x)
Retorna a parte inteira de **x**

double round(double x)
Retorna o valor arredondado de **x**
 round(10.5) → 11.0
 round(10.4) → 10.0

double ceil(double x)
Arredonda **x** para cima
 ceil(10.1) → 11.0

double floor(double x)
Arredonda **x** para baixo
 floor(10.9) → 10.0

Biblioteca stdlib.h

A biblioteca stdlib.h nos fornece várias funções úteis para manipulação de memória, geração de números aleatórios, execução de comandos no sistema, etc.

Geração de Números aleatórios