



INSTITUTO FEDERAL
PARANÁ

LÓGICA DE PROGRAMAÇÃO

PROF^a. M.Sc. JULIANA H Q BENACCHIO

Comandos de Decisão

- A maioria dos programas tomam decisões que afetam seu fluxo.
- Até agora, os programas eram totalmente sequenciais, ou seja, todos os passos eram executados em sequencia, sem nenhum tipo de modificação no fluxo do programa.
- Os comandos que tomam essas decisões são chamados de Estruturas de Controle ou ainda comandos de controle.

- A primeira estrutura de controle é a **Estrutura de Seleção**, que permite selecionar os passos que devem ser executados pelo programa em um determinado ponto.
- Esta estrutura também é chamada de **Estrutura de Decisão** ou **Estrutura Condicional**.
- Então, sempre que precisarmos tomar uma decisão em algum ponto do programa, devemos utilizar uma estrutura de seleção.

Estrutura de Seleção

- A seleção dos passos, que devem ou não ser executados, é feita a partir do resultado de uma expressão lógica ou relacional.
- Na prática, isto representa dotar o algoritmo de um mecanismo que lhe permita tomar decisões em tempo real, buscando atender a critérios preestabelecidos.

Estrutura de Seleção Simples

- A Estrutura de Seleção Simples permite definir um bloco de instruções que serão executadas apenas se forem atendidos os critérios definidos.
- Esta estrutura também é conhecida como desvio condicional simples.
- Na linguagem C, a estrutura de seleção simples é representada pelo comando **if** (SE)

Estrutura de Seleção Simples

- A sintaxe do **if** no C é a seguinte:

```
if (expressãoCondicional) {  
    codigo;  
}
```

- Uma expressão condicional é uma expressão cujo valor pode ser falso ou verdadeiro.
- Em C, falso é representado por **0** e verdadeiro é representado por **1**.

Estrutura de Seleção Simples

```
if (condição)  
comando1;
```

```
if (condição)  
{  
comando1;  
comando2;  
}
```

- A condição deve estar entre parênteses
- Se a condição for verdadeira (**true-1**) os comandos serão executados
- Se a condição for falsa (**false-0**) nada será executado

Estrutura de Seleção Simples

```
if (condição)  
    comando1;
```

```
if (condição)  
{  
    comando1;  
    comando2;  
}
```

- O **if** normalmente espera somente uma instrução no seu corpo.
- Para incluir várias instruções no corpo de um **if**, inclua as instruções dentro de chaves (**{** e **}**).
- Um conjunto de instruções contido dentro de um par de chaves é chamado de bloco

Operadores Relacionais

- Os operadores relacionais são utilizados em expressões condicionais para a comparação do valor de duas expressões:
 - $>$ → Maior que
 - $>=$ → Maior ou igual à
 - $<$ → Menor que
 - $<=$ → Menor ou igual à
 - $==$ → Igual à
 - $!=$ → Diferente de

Estrutura de Seleção Simples

Exemplo: Ler a idade de uma pessoa e mostrar se é maior de idade

Estrutura de Seleção Simples

```
int main()
{
    int idade;
    printf("Digite a idade: ");
    scanf("%d", &idade);
    if (idade >= 18)
        printf("Maior de idade\n");
    return 0;
}
```

Estrutura de Seleção Simples

- E se precisar indicar o comportamento que deve ser executado no caso da expressão condicional ser falsa?

- A Estrutura de Seleção Composta permite definir dois blocos de instruções, sendo que um deles será executado e o outro não, de acordo com o atendimento ou não dos critérios definidos.
- Esta estrutura também é conhecida como desvio condicional composto.
- Na linguagem C, a estrutura de seleção composta é representada pelo comando **if-else** (SE-SENÃO)

- A sintaxe do **if-else** no C é a seguinte:

```
if (expressãoCondiciona1) {  
    codigo1;  
}  
else{  
    codigo2;  
}
```

Estrutura de Seleção Composta

```
if (condição) {  
    comando1;  
    comando2;  
}  
else {  
    comandoA;  
    comandoB;  
}
```

- Se a condição for verdadeira (**true-1**) os comandos 1 e 2 serão executados
- Se a condição for falsa (**false-0**) os comandos A e B serão executados

Estrutura de Seleção Composta

```
int main()
{
    int idade;
    printf("Digite a idade: ");
    scanf("%d", &idade);
    if (idade >= 18)
        printf("Maior de idade\n");
    else
        printf("Menor de idade\n");
    return 0;
}
```

Estrutura de Seleção Composta

Exemplo: Ler um número e mostrar se é par ou ímpar

Estrutura de Seleção Composta



```
int main()
{
    int num, resto;
    printf("Digite um numero inteiro: ");
    scanf("%d", &num);
    resto = num % 2;
    if (resto == 0)
        printf("PAR\n");
    else
        printf("IMPAR\n");
    return 0;
}
```

Estrutura de Seleção Composta

Exemplo: Ler o sexo de uma pessoa e mostrar se é Feminino ou Masculino

Estrutura de Seleção Composta

```
int main()
{
    char sexo;
    printf("Digite o sexo: ");
    scanf("%c", &sexo);
    if (sexo == 'F')
        printf("Feminino\n");
    else
        printf("Masculino\n");
    return 0;
}
```

Funções para caracteres

#include <ctype.h>

toupper(char) – modifica para maiúsculo

tolower(char) – modifica para minúsculo

Funções para caracteres

```
int main()
{
    char sexo;
    printf("Digite o sexo: ");
    scanf("%c", &sexo);
    sexo = toupper(sexo);
    if (sexo == 'F')
        printf("Feminino\n");
    else
        printf("Masculino\n");
    return 0;
}
```

- Muitas vezes, dentro de um fluxo condicional, será necessário tomar uma nova decisão.
- Nesse caso podemos utilizar estruturas de seleção aninhadas, que nada mais são do que uma estrutura de seleção dentro de outra.
- Atenção: A cada novo nível de instruções, avançar na endentação, a fim de facilitar a leitura e o entendimento do algoritmo.

if Aninhados

```
if (condição1)
{
    if (condição2)
    {
        comando1;
        comando2;
    }
}
```

if-else Aninhados

```
if (condição1){  
    comando1;  
}else {  
    if (condição2)  
        comandoA;  
    else  
        comandoB;  
}
```

if - else Aninhados

- Não há um limite para o número de estruturas que podem ser aninhadas em um algoritmo, mas deve-se utilizar o bom senso a fim de se evitar algoritmos excessivamente longos e complexos.
- Pode-se dizer que, geralmente, quando as estruturas de seleção de um algoritmo atingem muitos níveis de aninhamento, o algoritmo não está utilizando a melhor opção possível de implementação.

if-else Aninhados

```
if (media >= 9)
    printf("A");
else
    if (media >= 8)
        printf("B");
    else
        if (media >= 7)
            printf("C");
        else
            printf("D");
```

if-else Aninhados

```
if (media >= 9)
    printf("A");
else if (media >= 8)
    printf("B");
else if (media >= 7)
    printf("C");
else
    printf("D");
```

if Aninhados e Operadores Lógicos

```
int main(){
    int num;
    printf("Digite um numero inteiro: ");
    scanf("%d", &num);
    if (num <= 9)
        printf("Numero com 1 digito\n");
    if (num >=10)
        if (num <= 99)
            printf("Numero com 2 digitos\n");
    if (num >= 100)
        printf("Numero com mais de 2 digitos\n");
    return 0;
}
```

- Os operadores lógicos são utilizados para conectar expressões lógicas sendo geralmente utilizados em expressões condicionais:

&& → AND (E lógico)

|| → OR (OU lógico)

! → NOT (Operador de negação)

Operador Lógico AND (&&)

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

Operador Lógico AND (&&)

```
int main(){
    int num;
    printf("Digite um numero inteiro: ");
    scanf("%d", &num);
    if (num <= 9)
        printf("Numero com 1 digito\n");
    if (num >=10 && num <= 99)
        printf("Numero com 2 digitos\n");
    if (num >= 100)
        printf("Numero com mais de 2 digitos\n");
    return 0;
}
```

Operador Lógico AND (&&)

```
int main(){
    int num;
    printf("Digite um numero inteiro: ");
    scanf("%d", &num);
    if (num <= 9)
        printf("Numero com 1 digito\n");
    else if (num >=10 && num <= 99)
        printf("Numero com 2 digitos\n");
    else if (num >= 100)
        printf("Numero com mais de 2 digitos\n");
    return 0;
}
```

Operador Lógico OR (| |)

A	B	A B
0	0	0
0	1	1
1	0	1
1	1	1

Operador Lógico OR (| |)

- Exemplo: Ler o sexo de uma pessoa e verificar se é Feminino ou Masculino

Operador Lógico OR (||)

```
int main()
{
    char sexo;
    printf("Digite o sexo: ");
    scanf("%c", &sexo);
    if (sexo == 'F' || sexo == 'f')
        printf("Feminino\n");
    else
        printf("Masculino\n");
    return 0;
}
```

Operador Lógico NOT (!)

A	!A
0	1
1	0

Funções matemáticas

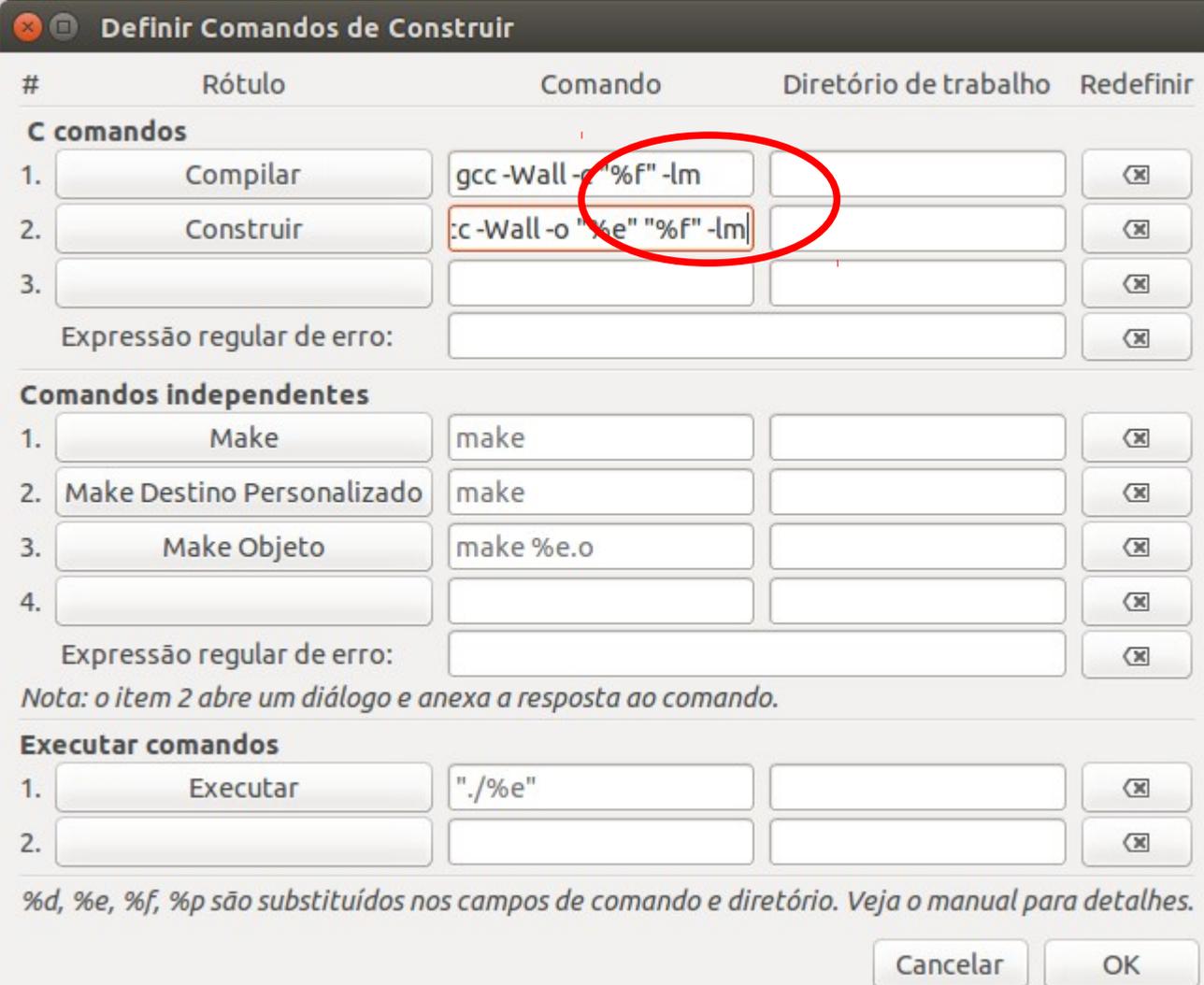
```
#include <math.h>
```

`sqrt(num)` – raiz quadrada de num

`pow(base, exp)` – base^{exp}

Funções matemáticas

- Compilando no Geany: incluir a opção **-lm**



Definir Comandos de Construir

#	Rótulo	Comando	Diretório de trabalho	Redefinir
C comandos				
1.	Compilar	gcc -Wall -c "%f" -lm		<input type="button" value="✕"/>
2.	Construir	cc -Wall -o "%e" "%f" -lm		<input type="button" value="✕"/>
3.				<input type="button" value="✕"/>
Expressão regular de erro:				<input type="button" value="✕"/>
Comandos independentes				
1.	Make	make		<input type="button" value="✕"/>
2.	Make Destino Personalizado	make		<input type="button" value="✕"/>
3.	Make Objeto	make %e.o		<input type="button" value="✕"/>
4.				<input type="button" value="✕"/>
Expressão regular de erro:				<input type="button" value="✕"/>
<i>Nota: o item 2 abre um diálogo e anexa a resposta ao comando.</i>				
Executar comandos				
1.	Executar	"./%e"		<input type="button" value="✕"/>
2.				<input type="button" value="✕"/>
<i>%d, %e, %f, %p são substituídos nos campos de comando e diretório. Veja o manual para detalhes.</i>				
				<input type="button" value="Cancelar"/> <input type="button" value="OK"/>