

Introdução à linguagem C

Linguagem de programação

Prof. Allan Rodrigo Leite

Unidades computacionais

- *BIT* (BInary DiGiT) – dígito binário
 - Menor unidade de informação
 - Armazena somente um valor 0 ou 1
- *Byte* (BinarY TErm) – termo binário
 - Conjunto de 8 bits
 - Pode representar números, letras, símbolos, imagens, etc.
- Palavra (*Word*)
 - É a quantidade de bits que a CPU pode processar por vez
 - Atualmente são comuns palavras de 32 ou 64 bits

Unidades computacionais

- Representação de uma memória de 1KB
 - No *byte* do endereço 2 está armazenado um dado
 - Neste caso, o dado representa o caractere A
 - O processador acessa o conteúdo de um *byte* a partir do endereço dele

Endereço	Byte							
0								
1								
2	0	1	0	0	0	0	0	1
...								
1023								

Unidades computacionais

- Os computadores apenas operam com *bytes* e palavras
 - Todo dado armazenado e processado é um conjunto de *bits* e *bytes*
 - Letras, dígitos e símbolos
 - Cores, ícones, figuras e fotos
 - Textos, músicas e vídeos
- Tabela ASCII
 - Tabela de códigos que representam textos em computadores
 - Cada código entre 0 e 255 representa um símbolo
 - <https://theasciicode.com.ar>

Arquivo executável

- Um arquivo executável é composto por milhares de instruções
 - São definidas por meio de sequências de 0 e 1
 - Cada instrução possui um significado para o processador
 - As instruções são chamadas de *opcode*
 - São formadas por um conjunto de *bytes*
- Linguagem de máquina
 - Também chamada de linguagem de baixo nível
 - Representa a estrutura de um arquivo executável é denominada
 - Para programar nesta linguagem é necessário conhecer os *opcodes* e operandos suportados pelo processador

Linguagem de máquina

- *Opcodes* são operações básicas que podem ser executadas pelo hardware de forma eficiente
 - Operações de processamento de dados
 - Operações matemáticas
 - Operações lógicas
 - Acessar variáveis ou estruturas de dados
 - Transferir dados da memória principal para o processador
 - Transferir bytes da entrada e saída para memória principal
 - Controlar a sequência (fluxo) do programa
 - Realizar desvios condicionais e incondicionais

Linguagem de máquina

- Formato básico de instruções de um processador



- *Opcode* (instrução)
 - Sequência de *bits* que identifica unicamente cada operação
- Operandos
 - Podem ter nenhum, 1, 2 ou 3 campos de *bits*
 - Determinam a origem dos dados utilizados na operação
 - Dependendo do *opcode*
 - Registrador ou endereço de memória depende do modo de endereçamento

Linguagem de montagem

- Notação legível para humanos para representar códigos de máquina
 - Os *opcodes* são representados por mnemonics
 - Auxiliam a memorização das instruções do processador
 - Também é conhecida como *Assembly*
- Tradução da linguagem de montagem para linguagem de máquina
 - É realizado por um programa chamado montador
 - O montador depende da arquitetura de processador

Linguagem de montagem

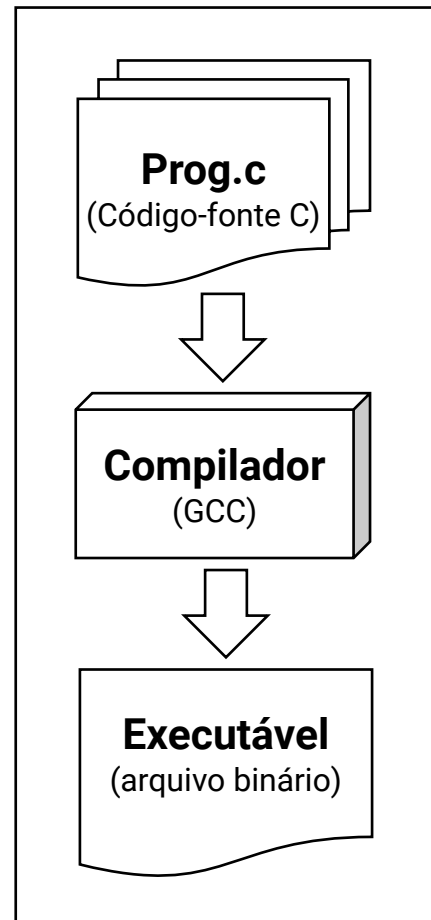
ADD	<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					0	0	0	1					<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>D</td><td>R</td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					D	R							<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>S</td><td>R</td><td>1</td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					S	R	1						<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>0</td><td></td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					0								<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>0</td><td>0</td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					0	0							<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>S</td><td>R</td><td>2</td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					S	R	2					
0	0	0	1																																																																											
D	R																																																																													
S	R	1																																																																												
0																																																																														
0	0																																																																													
S	R	2																																																																												
ADD	<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					0	0	0	1					<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>D</td><td>R</td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					D	R							<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>S</td><td>R</td><td>1</td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					S	R	1						<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>1</td><td></td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					1								<table><tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr><tr><td colspan="6">imm5</td></tr><tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr></table>								imm5																	
0	0	0	1																																																																											
D	R																																																																													
S	R	1																																																																												
1																																																																														
imm5																																																																														
AND	<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					0	1	0	1					<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>D</td><td>R</td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					D	R							<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>S</td><td>R</td><td>1</td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					S	R	1						<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>0</td><td></td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					0								<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>0</td><td>0</td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					0	0							<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>S</td><td>R</td><td>2</td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					S	R	2					
0	1	0	1																																																																											
D	R																																																																													
S	R	1																																																																												
0																																																																														
0	0																																																																													
S	R	2																																																																												
AND	<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					0	1	0	1					<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>D</td><td>R</td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					D	R							<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>S</td><td>R</td><td>1</td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					S	R	1						<table><tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr><tr><td colspan="6">imm5</td></tr><tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr></table>									imm5																													
0	1	0	1																																																																											
D	R																																																																													
S	R	1																																																																												
imm5																																																																														
NOT	<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					1	0	0	1					<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>D</td><td>R</td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					D	R							<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>S</td><td>R</td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					S	R							<table><tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr><tr><td colspan="6">111111</td></tr><tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr></table>									111111																													
1	0	0	1																																																																											
D	R																																																																													
S	R																																																																													
111111																																																																														
JMP	<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					1	1	0	0					<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>0</td><td></td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					0								<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td>0</td><td>0</td><td></td><td></td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					0	0							<table><tr><td> </td><td> </td><td> </td><td> </td></tr><tr><td colspan="4">BaseR</td></tr><tr><td> </td><td> </td><td> </td><td> </td></tr></table>					BaseR								<table><tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr><tr><td colspan="6">000000</td></tr><tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr></table>								000000																	
1	1	0	0																																																																											
0																																																																														
0	0																																																																													
BaseR																																																																														
000000																																																																														

Linguagem de programação

- As linguagens de programação de alto nível impulsionam o desenvolvimento de software desde o surgimento dos computadores
 - É uma abstração para instruções complexas em linguagem de máquina
 - Primeira linguagem de programação surgiu na década de 50
 - FORTRAN (FORmula TRANslation System), criada pela IBM em 1954
 - Lista das principais linguagens de programação
 - <https://www.tiobe.com/tiobe-index>
- Instruções em linguagens de programação de alto nível são escritas de forma muito mais clara e legível para o desenvolvedor

Compilador

- Como o computador compreende um programa desenvolvido em linguagem de alto nível?
 - É necessário um processo de tradução da linguagem de alto nível para linguagem de montagem
 - Este processo de tradução é conhecido por compilação
- Mesmo assim, o programador precisa seguir uma série de regras ao utilizar uma linguagem de alto nível



```
void swap(int v[], int k) {  
    int aux;  
    temp = v[k];  
    v[k] = v[k + 1];  
    v[k + 1] = aux;  
}
```

Código-fonte em linguagem
de alto nível (C)

Compilador

```
swap:  
    muli    $2, $5, 2  
    add     $2, $5, 4  
    lw      $15, 0($2)  
    lw      $16, 4($2)  
    sw      $16, 0($2)  
    sw      $15, 4($2)  
    jr      $31
```

Programa em
assembly (MIPS)

Montador

```
0000000001010000100000000000011000  
000000000000110000001100000100001  
1000110001100010000000000000000000  
1000110011110010000000000000000100  
1010110011110010000000000000000000  
1010110001100010000000000000000100  
0000001111100000000000000000001000
```

Programa binário em linguagem de
máquina (MIPS)

Compilador

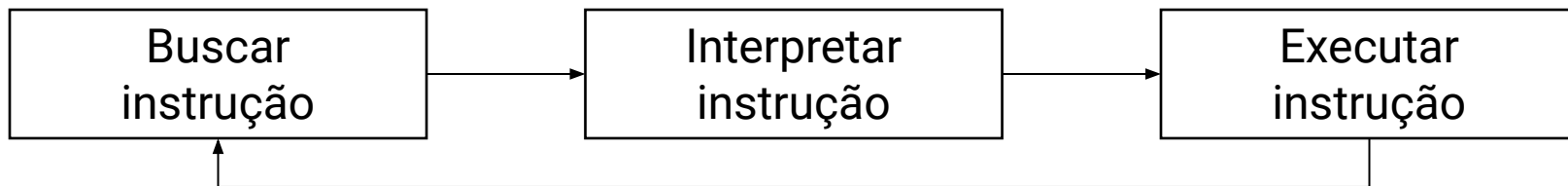
- Etapas executadas pelo compilador
 - Análise léxica
 - Decompõe o código fonte em seus elementos individuais distintos
 - Verifica se estes elementos estão de acordo com as regras da linguagem
 - Por exemplo: comandos, operadores, variáveis
 - Análise sintática
 - Conjunto de regras que definem como uma linguagem pode ser utilizada
 - O desenvolvedor precisa seguir estas regras ao escrever o programa
 - Do contrário, o compilador não conseguirá traduzi-las

Compilador

- **Análise léxica e sintática**
 - Quando o código-fonte contém algum erro de sintaxe, o compilador
 - Interrompe o processo de tradução
 - Indica a linha do arquivo onde o erro ocorreu
 - Nestes casos o arquivo executável não é gerado
 - Exemplos de erros de sintaxe
 - Palavras com erros de grafia
 - Parênteses ou aspas que não balanceadas
 - Ausência de vírgulas, pontos ou ponto e vírgula

Execução de um arquivo executável

- O processador é um dispositivo que opera em ciclos
 - Em cada ciclo, o processador executa uma instrução
 - Instruções são pequenas tarefas executadas sobre operandos
- Representação de um ciclo de operação



Sistema operacional

- Programa que controla e coordena todas as operações básicas em um computador
 - Inclui a execução de outros programas
- Oferece aos programas funções como
 - Controle de entrada e saída de dados
 - Alocação de memória
 - Gerenciamento de dados
 - Processamento das instruções do programa

Linguagem de programação C

- Linguagem de programação a ser utilizada nesta disciplina
 - Amplamente utilizada comercialmente e no meio acadêmico
 - Sucesso do sistema operacional Unix auxiliou na popularização
 - A linguagem C é considerada simples e extremamente flexível
- A base do C foi desenvolvida entre os anos 1969 e 1973
 - Em paralelo com o desenvolvimento do Unix
 - Maiores contribuições ocorreram em 1972

Linguagem de programação C

- Toda linguagem de programação deve seguir uma sintaxe
 - São regras detalhadas para construção de uma instrução válida
- Estrutura básica de um programa em C
 - Possui uma função chamada `main`
 - Representa a primeira função do programa a ser executada

```
int main() {  
    return 0;  
}
```

Linguagem de programação C

- Para compilar, usa-se o programa gcc
 - gcc <código-fonte> -o <arquivo executável>
- Exemplo
 - gcc teste.c -o teste

Linguagem de programação C

- Olá Mundo

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

```
int main() {  
    printf("Ola mundo.");  
  
    return 0;  
}
```

Variáveis e constantes

- Variável
 - Espaço na memória para armazenar um dado
 - Não é uma variável no sentido matemático
- Uma variável possui
 - Identificador
 - Nome exclusivo para identificar e acessar o espaço de memória
 - Tipo
 - Define a natureza do dado
 - Escopo
 - Determina onde (local) a variável pode ser acessada

Variáveis e constantes

- Tipos de dados

Tipo	Tamanho	Menor valor	Maior valor
char	1 byte	-128	+127
unsigned char	1 byte	0	+255
short int (short)	2 bytes	-32.768	+32.767
unsigned short int	2 bytes	0	+65.535
int (*)	4 bytes	-2.147.483.648	+2.147.483.647
long int (long)	4 bytes	-2.147.483.648	+2.147.483.647
unsigned long int	4 bytes	0	+4.294.967.295
float	4 bytes	-10^{38}	$+10^{38}$
double	8 bytes	-10^{308}	$+10^{308}$

(*) depende da máquina, sendo 4 bytes para arquiteturas de 32 bits

Variáveis e constantes


- Declaração de variável
 - Devem ser explicitamente declaradas
 - Podem ser declaradas em conjunto
 - Somente armazenam valores do mesmo tipo com que foram declaradas

```
char a;    /*declara uma variável do tipo char*/  
int b;     /*declara uma variável do tipo int*/  
float c;   /*declara uma variável do tipo float*/  
int d, e;  /*declara duas variáveis do tipo int*/
```

Variáveis e constantes

- Quando uma variável é declarada, seu valor inicial não é modificado e seu conteúdo é desconhecido
 - Comumente estes valores iniciais desconhecidos são chamados de lixo

```
int main() {  
    int a;  
    int b;  
    b = a;  
    a = 10;  
    return 0;  
}
```




	0	1	2	3	4	5	6	7
...	0	1	1	1	0	0	1	0
1712	1	1	0	0	1	1	1	1
a	0	0	0	0	1	1	0	0
	0	1	1	0	0	1	0	1
	1	0	0	1	0	0	0	0
	0	0	0	0	1	0	1	0
1717	1	1	1	1	1	0	1	1
...	1	0	0	1	0	1	0	0

Variáveis e constantes

- Para evitar problemas, recomenda-se que nenhuma variável deve ser utilizada antes de ser inicializada
 - b recebe 1100011001011001000000001010 (em decimal, 207982602)
 - Em outra execução, o valor pode ser outro

```
int main() {  
    int a;  
    int b;  
    b = a;  
    a = 10;  
    return 0;  
}
```



A large arrow points from the variable 'a' in the code to the memory layout table.

	0	1	2	3	4	5	6	7
...	0	1	1	1	0	0	1	0
1712	1	1	0	0	1	1	1	1
a	0	0	0	0	1	1	0	0
	0	1	1	0	0	1	0	1
	1	0	0	1	0	0	0	0
	0	0	0	0	1	0	1	0
1717	1	1	1	1	1	0	1	1
...	1	0	0	1	0	1	0	0

Variáveis e constantes

- Constante
 - Valor armazenado na memória
 - Possui um tipo, indicado pela sintaxe

123 /*constante inteira do tipo int*/

12.45 /*constante real do tipo double*/

1245e-2 /*constante real do tipo double*/

12.45F /*constante real do tipo float*/

Expressões

- Combinação de variáveis, constantes e operadores que, quando avaliada, produz algum valor
- Tipos de expressões
 - Atribuição
 - Variável recebe um determinado valor
 - Expressão aritmética, incremento ou decremento
 - Resulta em um número (inteiro ou real)
 - Expressão lógica ou relacional
 - Resulta em VERDADEIRO ou FALSO

Expressões aritméticas

Tipo	Operador	Descrição	Inteiros	Reais
Unário	–	Sinal negativo	–2	–2.0
			–a	–b
Binário	+	Adição	a + 2	b + 2.0
	–	Subtração	a – 2	b – 2.0
	*	Multiplicação	a * 2	b * 2.0
	/	Divisão	a / 2	b / 2.0
	%	Módulo	a % 2	Operação não definida para reais

Operadores e expressões

- Operadores de atribuição ($=$, $+=$, $-=$, $*=$, $/=$, $\%=$)
 - A atribuição é tratada como uma expressão
 - A ordem é da direita para a esquerda
 - Também oferece uma notação compacta para atribuições em que a mesma variável aparece nos dois lados

```
i += 2;      //é equivalente a i = i + 2  
x *= y + 1;  //é equivalente a x = x * (y + 1)
```

Operadores e expressões

- Operadores de incremento e decremento (++ , --)
 - Incrementa ou decrementa uma unidade de valor de uma variável
 - Estes operadores não se aplicam a expressão
 - O incremento/decremento pode ocorrer antes ou depois do uso da variável

`n++; //incrementa uma unidade em n, depois de ser usado`

`++n; //incrementa uma unidade em n, antes de ser usado`

`n = 5;`

`x = n++;`

`x = ++n;`

`a = 3;`

`b = a++ * 2;`

Operadores e expressões

- Operadores de incremento e decremento (++ , --)
 - Incrementa ou decrementa uma unidade de valor de uma variável
 - Estes operadores não se aplicam a expressão
 - O incremento/decremento pode ocorrer antes ou depois do uso da variável

`n++; //incrementa uma unidade em n, depois de ser usado`

`++n; //incrementa uma unidade em n, antes de ser usado`

`n = 5;`

`x = n++; //x recebe 5 e n é incrementado para 6`

`x = ++n; //n é incrementado para 7 e x recebe 7`

`a = 3;`

`b = a++ * 2; //a é incrementado para 4 e b recebe 6`

Operadores e expressões

- Operadores relacionais (<, <=, ==, >=, >, !=)
 - O resultado será 0 ou 1
 - Equivalente a FALSO (igual a 0) ou VERDADEIRO (diferente de 0)

```
int a, b;  
int c = 23;  
int d = c + 4;
```

```
c < 20  
d > c
```


Operadores e expressões

- Operadores relacionais (<, <=, ==, >=, >, !=)
 - O resultado será 0 ou 1
 - Equivalente a FALSO (igual a 0) ou VERDADEIRO (diferente de 0)

```
int a, b;  
int c = 23;  
int d = c + 4;
```

```
c < 20 //retorna 0  
d > c  //retorna 1
```

Operadores e expressões

- Operadores lógicos (&&, ||, !)
 - A avaliação ocorre da esquerda para a direita
 - A avaliação para quando o resultado for conhecido, antes mesmo de completar a expressão

```
int a, b;  
int c = 23;  
int d = c + 4;
```

```
a = (c < 20) || (d > c);  
b = (c < 20) && (d > c);
```

Operadores e expressões

- Operadores lógicos (&&, ||, !)
 - A avaliação ocorre da esquerda para a direita
 - A avaliação para quando o resultado for conhecido, antes mesmo de completar a expressão

```
int a, b;  
int c = 23;  
int d = c + 4;
```

```
a = (c < 20) || (d > c); //1 e as duas expressões são validadas  
b = (c < 20) && (d > c); //0 e só a primeira expressão é validada
```

Operadores e expressões

- Função `sizeof`
 - Retorna o número de bytes ocupados por um tipo de dados

```
int a = sizeof(float); //armazena 4 em a
```

- Conversão de tipo de dados
 - Conversão de tipo é automática na avaliação de uma expressão
 - Conversão de tipo pode ser requisitada explicitamente

```
float f;  
float f = 3;           //f recebe 3.0F, conversão implícita  
int g, h;  
g = (int) 3.5;         //3.5 é convertido (e arredondado) para int  
h = (int) 3.5 % 2;     //conversão realizada antes do módulo
```

Entrada e saída

- Função `printf`
 - Possibilita a saída de valores conforme o formato especificado

```
printf(formato, expr1, expr2, ..., exprN);
```

```
printf("%d %g", 33, 5.3);  
//imprimirá na console a linha "33 5.3"
```

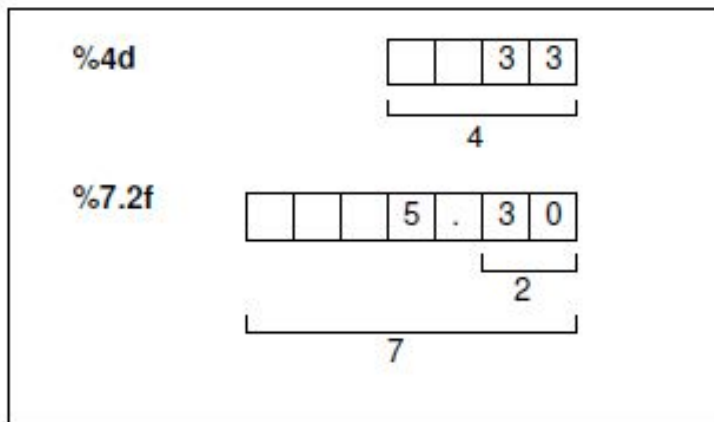
```
printf("Inteiro = %d Real = %g", 33, 5.3);  
//imprimirá na console a linha "Inteiro = 33 Real = 5.3"
```

Entrada e saída

- Formatos do `printf`
 - `%c` especifica um `char`
 - `%d` especifica um `int`
 - `%u` especifica um `unsigned int`
 - `%f` especifica um `double` ou `float`
 - `%e` especifica um `double` ou `float` em formato científico
 - `%g` especifica um `double` ou `float` em formato mais apropriado (`%f` ou `%e`)
 - `%s` especifica uma cadeia de caracteres
- Caractere de escape
 - `\n` quebra de linha
 - `\t` tabulação
 - `\"` caractere `"`
 - `\\` caractere `\`

Entrada e saída

- Tamanhos de campo printf



Entrada e saída

- Função scanf
 - Captura valores fornecidos via teclado

```
scanf(formato, var1, var2, ..., varN);
```

```
int n;
```

```
scanf("%d", &n);
```

```
//valor inteiro digitado pelo usuário é armazenado em n
```


Entrada e saída

- Formatos do scanf
 - %c especifica um char
 - %d especifica um int
 - %u especifica um unsigned int
 - %f, %e, %g especifica um float
 - %lf, %le, %lg especifica um double
 - %s especifica uma cadeia de caracteres
- Caracteres diferentes dos especificadores
 - Servem para cercar a entrada

```
scanf("%d:%d", &hora, &minuto);
```

Controle de fluxo

- Instrução `if`
 - Comando básico para definir desvios ou tomada de decisão
 - Se condição `<expr>` for verdadeira (diferente de 0), executa `<bloco 1>`
 - Se condição `<expr>` for falsa (igual a 0), executa o `<bloco 2>`
 - O bloco `else` é opcional

```
if (<expr>) {  
    <bloco 1>  
}  
[ else {  
    <bloco 2>  
} ]
```

Controle de fluxo

- Estrutura de bloco
 - Declaração de variáveis
 - Só podem ocorrer no início do corpo da função ou bloco
 - Esta restrição não existe em versões mais recentes do C (C99)
 - Escopo de uma variável
 - Uma variável declarada dentro de um bloco é válida apenas no próprio bloco
 - Após o término da execução do bloco, a variável deixa de existir

```
if (n > 0) {  
    int i;  
    ...  
}  
//a variável i não existe mais
```

Controle de fluxo

- Instrução switch
 - Seleciona um bloco entre várias opções
 - A instrução break evita que as opções subjacentes sejam executadas
 - O bloco default é executado quando nenhuma opção for válida

```
switch (<expr>) {  
    case <opção 1>:  
        <bloco 1>;  
        break;  
    ...  
    default:  
        <bloco padrão>;  
        break;  
}
```

Laços de repetição

- Fatorial de número inteiro não negativo
 - $n! = n \times (n - 1) \times (n - 2) \times \dots$
- Cálculo não recursivo de `fatorial(n)`
 - Comece com $k = 1$ e $f = 1$
 - Faça enquanto $k \leq n$
 - f recebe $f * k$
 - Incrementa k em uma unidade

Laços de repetição

- Instrução `while`
 - Enquanto `<expr>` for verdadeira, o `<bloco>` é executado
 - Quando `<expr>` for falsa, o laço de repetição é encerrado

```
while (<expr>) {  
    <bloco de instruções>  
}
```

Laços de repetição

```
int main () {  
    int k, n;  
    int f = 1;  
    printf("Digite um numero inteiro nao negativo:");  
    scanf("%d", &n);  
    k = 1;  
    while (k <= n) {  
        f = f * k; /* f = f * k é equivalente a f *= k */  
        k = k + 1; /* k = k + 1 é equivalente a k++ */  
    }  
    printf("Fatorial = %d \n", f);  
    return 0;  
}
```

Laços de repetição

- Instrução for
 - Forma compacta para definir laços
 - Expressão inicial
 - Condição de parada
 - Expressão de incremento

```
for (<expr inicial>; <condição>; <expr incremento>) {  
    <bloco de instruções>  
}
```


Laços de repetição

```
int main () {  
    int k, n;  
    int f = 1;  
  
    printf("Digite um numero inteiro nao negativo:");  
    scanf("%d", &n);  
  
    for (k = 1; k <= n; k++) {  
        f *= k;  
    }  
  
    printf(" Fatorial = %d \n", f);  
    return 0;  
}
```

Laços de repetição

- Comando `do while`
 - Condição de parada é avaliada ao final do bloco
 - Portanto, sempre executará uma vez o bloco de repetição

```
do {  
    <bloco de instruções>  
} while (<condição>);
```

Laços de repetição

```
int main () {  
    int k, n;  
    int f = 1;  
    do {  
        printf("Digite um numero inteiro nao negativo:");  
        scanf("%d", &n);  
    } while (n < 0);  
    for (k = 1; k <= n; k++) {  
        f *= k;  
    }  
    printf(" Fatorial = %d \n", f);  
    return 0;  
}
```

Laços de repetição

- Comandos break e continue
 - break termina o laço de repetição
 - continue termina a iteração atual e vai para a próxima

```
for (int i = 0; i < 10; i++) {  
    if (i == 5) break;  
    printf("%d ", i);  
}
```

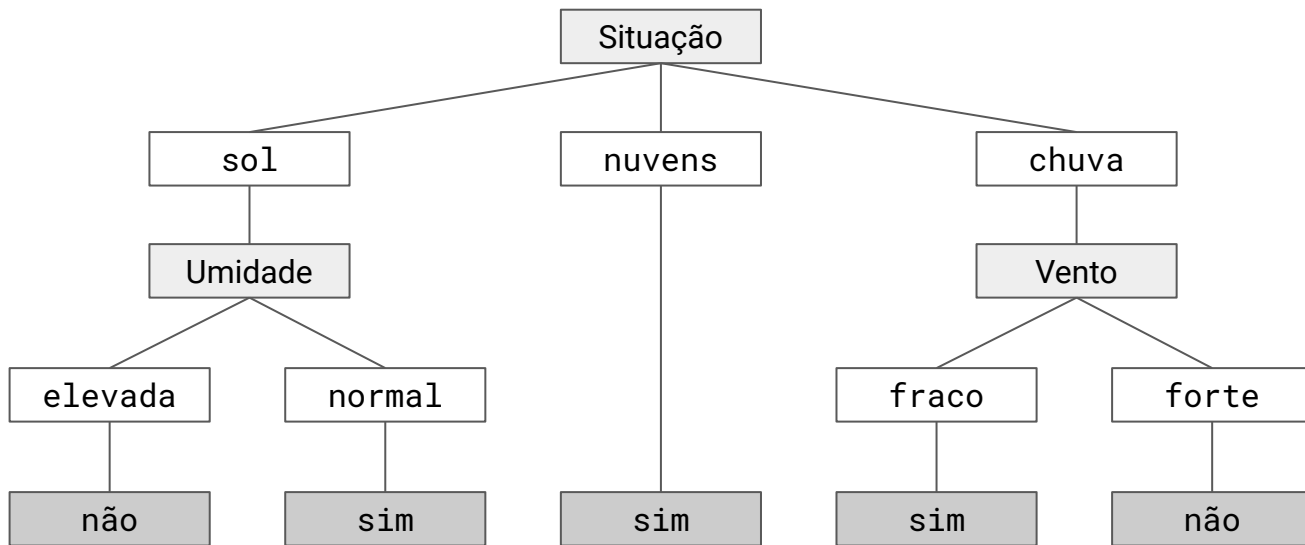
```
for (int i = 0; i < 10; i++) {  
    if (i == 5) continue;  
    printf("%d ", i);  
}
```

Exercícios

1. Faça um programa que solicite a idade de uma pessoa. Em seguida, o programa deve exibir uma mensagem informando se a pessoa é maior de idade. Utilize o tipo de dado mais conveniente e trate dados inválidos, tais como valor negativo.
 - Exemplo 1: dada a idade 19, a saída deve ser "Maior de idade".
 - Exemplo 2: dada a idade 12, a saída deve ser "Menor de idade".
2. Faça um programa que solicite o nome de uma substância e sua substância. Em seguida, o programa deve exibir uma mensagem informando o estado físico que a substância se encontra. Devem ser consideradas as substâncias água, etanol, ferro, ouro e mercúrio, bem como seus respectivos pontos de fusão e ebulição. Utilize os tipos de dado mais convenientes e trate dados inválidos, tais como substância desconhecida.
 - Exemplo 1: dada a substância água e a temperatura 58°, a saída esperada é "Estado líquido".
 - Exemplo 2: dada a substância etanol e a temperatura 80°, a saída esperada é "Estado gasoso".

Exercícios

3. Faça um programa que reproduza a árvore de decisão abaixo para indicar se as condições climáticas estão favoráveis ou não para jogar tênis. Utilize tipos de dado mais convenientes e trate situações de dados inválidos, bem como solicitar informações irrelevantes para um dado fluxo.
- Exemplo 1, ao informar a situação sol e umidade elevada, a saída esperada é não.
 - Exemplo 2, ao informar a situação nuvens, a saída esperada é sim.



Exercícios

4. Faça um programa que solicite o comprimento de três lados de um triângulo. Em seguida, o programa deve informar se o triângulo é equilátero, isósceles ou escaleno. Utilize os tipos de dados mais convenientes e trate dados inválidos, tais como valores negativos ou se os comprimentos não formam um triângulo.
- Exemplo 1, dado o lado $a = 4$, $b = 4$ e $c = 4$, a saída esperada é equilátero.
 - Exemplo 2, dado o lado $a = 5$, $b = 7$ e $c = 8$, a saída esperada é escaleno.
5. Faça um programa que solicite o salário de um funcionário no regime CLT. Em seguida, o programa deve informar a alíquota do Imposto de Renda e o salário bruto após a dedução do imposto. Utilize os tipos de dados mais convenientes e trate dados inválidos, tais como salário negativo. Além disso, considere também a parcela a deduzir do Imposto de Renda.
- Exemplo 1, dado o salário R\$ 1.500,00, a saída esperada é isento e R\$ 1.500,00.
 - Exemplo 2, dado o salário R\$ 3.500,00, a saída esperada é 15% e R\$ 3.329,80.
 - $3.500,00 - (3.500,00 * 0,15 - 354,80) = 3.329,80.$

Exercícios

6. Faça um programa que solicite um número positivo. Em seguida, o programa deve informar a tabuada de 0 a 10 deste número. Utilize os tipos de dados adequados e trate dados inválidos, tais como valores negativos.
 - Exemplo, dado o número 7, a saída esperada é " $0 \times 7 = 0$, $1 \times 7 = 7$, ..., $10 \times 7 = 70$ ".
7. Faça um programa que solicite um número positivo. Em seguida, o programa deve exibir os números pares de 0 até o número informado pelo usuário. Utilize os tipos de dados adequados e trate dados inválidos, tais como valores negativos.
 - Exemplo, dado o número 10, a saída é "2, 4, 6, 8, 10".
8. Faça um programa que solicite n números reais. Em seguida, o programa deve informar os dois maiores valores do conjunto de números informados.
 - Exemplo, dados os números 14, 3, 10, 2, 1, 8, 15 e 5, a saída esperada é "14 e 15".

Exercícios

9. A série de Fibonacci é uma sequência de números inteiros, começando por 1 e 1, onde cada termo subsequente corresponde à soma dos dois anteriores. Faça um programa que solicite o n -ésimo termo da série e, em seguida, informe o número correspondente ao termo.
 - Exemplo 1, dado o termo 7, a saída esperada é 13.
 - Exemplo 2, dado o termo 9, a saída esperada é 34.
10. Faça um programa que solicite n números reais e informe qual é o menor e maior valor dentre eles, bem como a média dos números.
 - Exemplo, dados os valores 14, 3, 10, 2, 1, 8, 15 e 5, a saída é "Maior=15, Menor=1, Média= 7,25".
11. Números primos são os números naturais que têm apenas dois divisores distintos: 1 e ele mesmo. Faça um programa que solicite um número natural, em seguida, informe o número é primo ou não.
 - Exemplo 1, dado o termo 23, a saída esperada é sim.
 - Exemplo 2, dado o termo 77, a saída esperada é não.

Exercícios

12. Faça um programa que gere um número aleatório entre 0 e 100 e o usuário deve tentar adivinhar o número gerado. Se o usuário errar o palpite, o programa deve informar se o número gerado é maior ou menor que o número informado e solicitar um novo palpite. Se o usuário acertar o palpite, o programa é encerrado e deve ser exibido o número de tentativas. Dica: utilize as funções `rand()` ou `srand()` para gerar os números aleatórios.
 - Exemplo, dado o número aleatório 7 e os palpites 100, 40, 20, 10 e 7, a saída esperada é 5.
13. Faça um programa que tente adivinhar um número aleatório entre 0 e 100 que o usuário mentalizou. Neste caso, o programa sugere um número e o usuário deve informar se o palpite é o número do usuário. Se o programa errar o palpite, o usuário deve informar se o número gerado é maior ou menor que o número informado. Se o programa acertar o palpite, o programa é encerrado e deve ser exibido o número de tentativas.
 - Exemplo, dado o número aleatório 7 e os palpites 50, 25, 12, 6, 9 e 7, a saída é 6.

Exercícios

14. Faça um programa que solicite a base e o expoente e, em seguida, exiba o valor da potência. Não utilize a função `pow()` para calcular a potência. Ao invés disso, utilize um laço de repetição para realizar este cálculo.
 - Exemplo 1, dado a base 4 e expoente 3, a saída esperada é 64.
 - Exemplo 2, dado a base 3 e expoente 1, a saída esperada é 3.
15. Faça um programa que solicite dois números positivos e, em seguida, exiba o produto destes números. Não utilize o operador aritmético `*`. Ao invés disso, utilize um laço de repetição para realizar este cálculo.
 - Exemplo, dado os números aleatório 7 e 3, a saída é 21.
16. Faça um programa que solicite dois números positivos e, em seguida, exiba o quociente inteiro da divisão desses números. Não utilize o operador aritmético `/`. Ao invés disso, utilize um laço de repetição para realizar este cálculo.
 - Exemplo, dado os números aleatório 19 e 3, a saída é 6.

Introdução à linguagem C

Linguagem de programação

Prof. Allan Rodrigo Leite