parte1 _ Tipos, Variáveis, Entrada e Saída

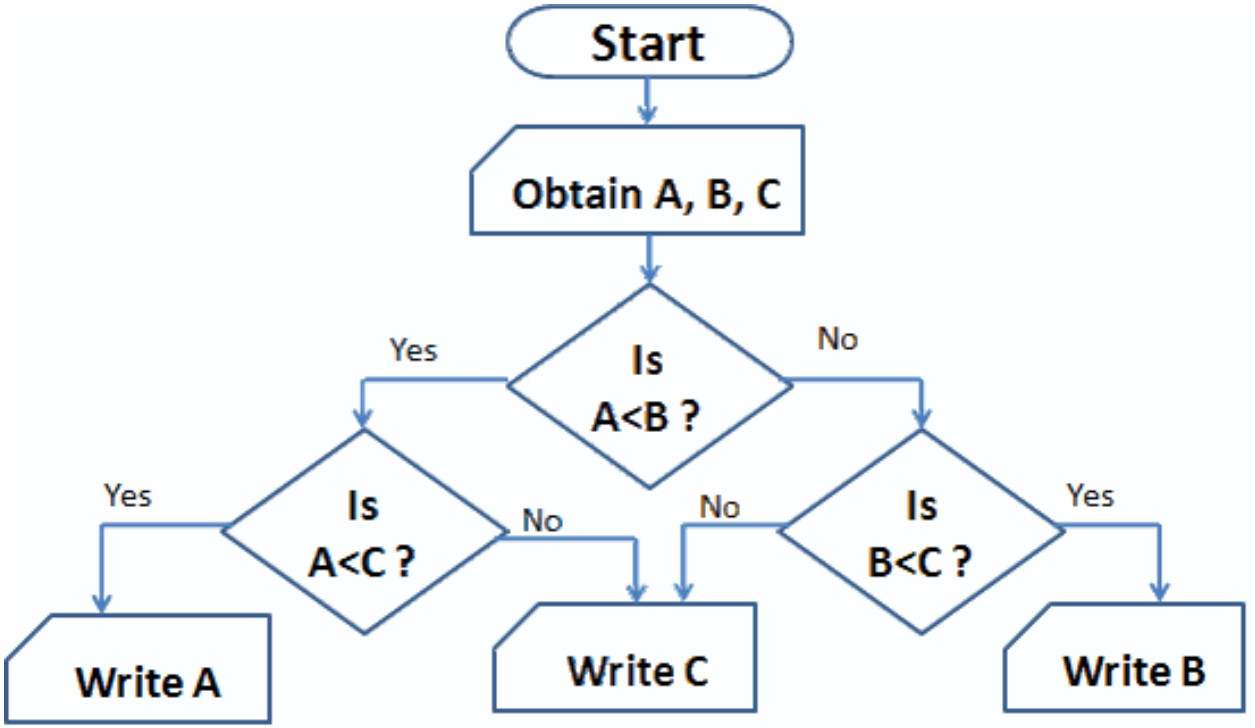
Marcos Silvano / DACOM

BCC32A-Algoritmos 1

Introdução

Programa de Computador (Software)

- Algoritmo
  - **Sequência de instruções** para realizar uma tarefa
- Exemplos de algoritmos
  - Algoritmo da Soma, Subtração, Multiplicação e Divisão que utilizamos
  - Chegar a um local seguindo intruções
  - Receita culinária e Manual de instruções de lego
  - Atividades de um funcionário de fábrica



Programas de Computador

- Programas são feitos por algoritmos escritos em linguagem de programação
  - Sistemas Operacionais: Windows, Linux e MacOS
  - Apps Android/iOS: Whatsapp, Facebook, CamScanner, AirDroid
  - Editores/ferramentas: Word, Exel, Gimp, GEdit, Notepad++
  - Serviços de e-mail: Gmail, Homail
  - Jogos: Mobile, Web, PC, Consoles de Videogames Linguagem de Programação possui
  - Sintaxe/forma bem definida
  - Conjunto de palavras e instruções pequeno

Linguagens: nível de abstração

- A linguagem utilizada depende da entidade que compreenderá e executará o algoritmo
- Pessoa: maior nível de abstração
  - Instruções mais complexas/abstratas: andar, pegar, ler, olhar, cozinhar, costurar
- Computador: baixo nível de abstração
  - Instruções bem mais simples: Copiar valor de uma posição de memória para outra, somar, multiplicar, verificar igualdade, negar

```
// exemplo de código em linguagem de programação
total = preco * lucro * taxas;
distancia = raiz_quadrada(quadrado(x2 - x1) + quadrado(y2 - y))
dano = forca_ataque * elemento_ataque + critico
```

Linguagens de Programação

- Existem centenas de linguagens de programação
  - C - 1972, influenciou quase todas as linguagens usadas hoje
  - C++ - versão melhorada de C (contém C)
    - ★ Possui mais recursos e facilidades
    - ★ Continua em evolução: C++11, C++14, C++17
  - Java - (Oracle) usada principalmente para desenvolvimento de apps Android
  - Swift - (Apple) usada para desenvolvimento de apps para MacOS e iOS
  - C# - (Microsoft) criada como resposta à crescente popularidade do Java. Ganhou muitos adeptos na comunidade de software open source
  - JavaScript - (Netscape/Mozilla) - usada para desenvolvimento de sistemas web, apps desktop e mobile
  - Python - usada principalmente para utilitários, scripts de SO e apps científicos

Hardware e Software

- CPU :: Central Processing Unit
  - Executa instruções sequencialmente do programa, que encontra-se na memória RAM.
  - Valores/operandos sendo trabalhos na instrução (ex: somando dois números) são carregados da memória para registradores.
    - ⋆ Usualmente a CPU realiza operações sobre dados que estão em seus registradores (são internos à CPU).
- Dado e memória em computador é mensurado em bits (0 ou 1)
  - Byte $\rightarrow$ 8 bits
  - Word $\rightarrow$ tamanho do dado que a CPU consegue representar
    - ⋆ Antes 16 bits... até ontem 32 bits... hoje 64 bits
  - Word $\rightarrow$ também influencia na quantidade de posições (Bytes) que consegue endereçar
    - ⋆ 32 bits $\rightarrow$ até 4 Giga Bytes (4 bilhões de bytes)
    - ⋆ 64 bits $\rightarrow$
  - Mais bits $\rightarrow$ maior precisão numérica (GPU) e mais possibilidades de endereços (CPU)

Linguagens de baixo nível

- **Código Binário**
  - Circuito eletrônico: 0 baixa voltagem / 1 alta voltagem
  - Compreendido e executado pelo computador
  - Um programa exe/bin é formado por texto que representa código binário
  - Existe uma estrutura específica para os executáveis de cada SO
- **Assembly** / Linguagem de montagem
  - Tradução quase direta para Código Binário

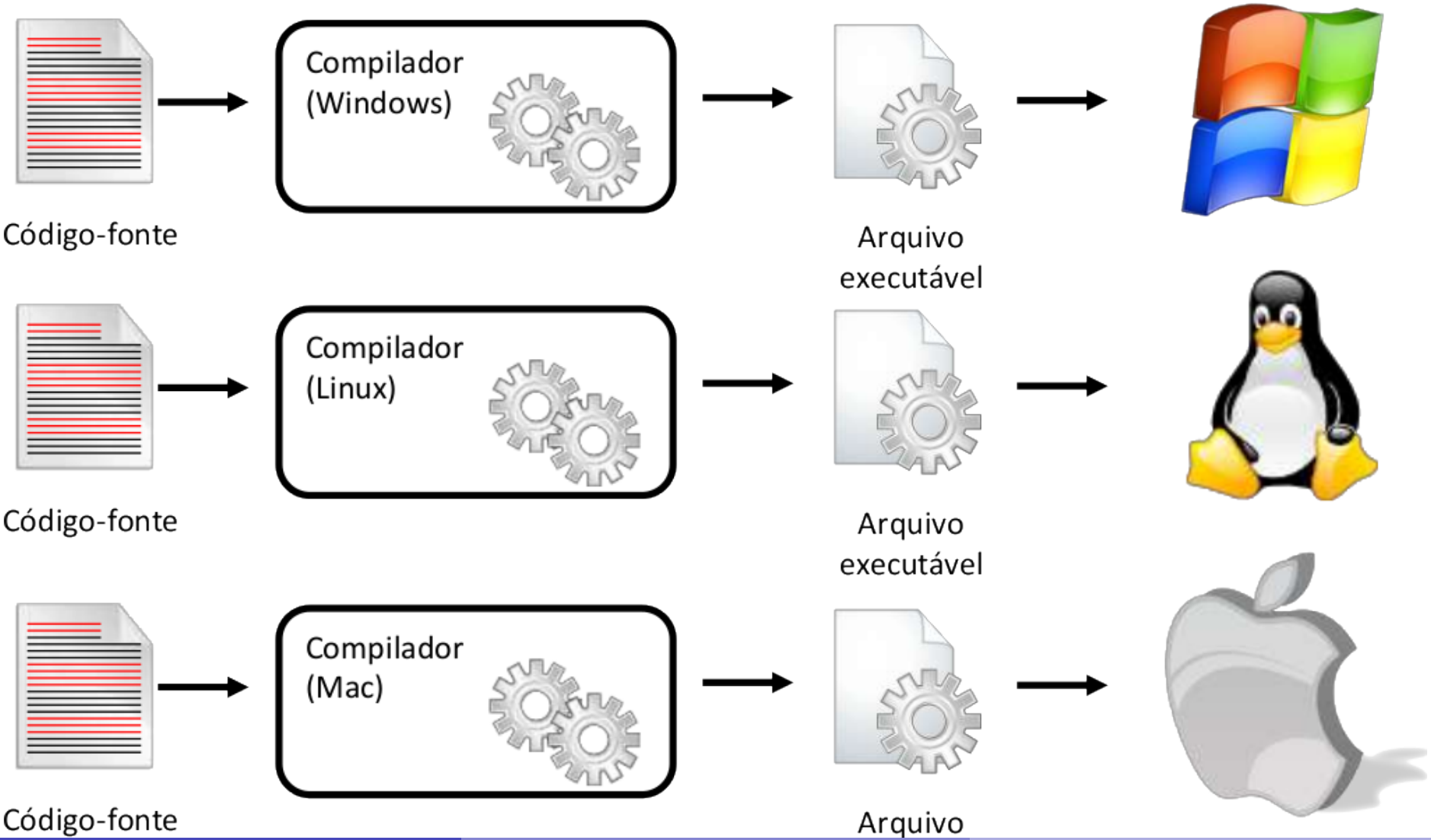
```
// Programa simples para somar A e B e armazenar em R
BEGIN
 Load  A //registrador AC recebe valor que está na posição de mem A
 Add   B //soma o valor que está em B ao registrador acumulador(AC)
 Store R //copia o valor de registrador AC para o endereço R
 Halt
// Posições de memória
 A, Dec 5
 B, Dec 4
 R, Dec 1
END
```

Linguagens de Alto Nível

- Linguagens populares de Alto Nível
  - C/C++, C#, Java, Swift
  - Compiladas: compilador transforma código-fonte em código binário para CPU
- Linguagens de mais alto nível
  - JavaScript, Python, Lua
  - Interpretadas: interpretador analisa e executa instruções, linha-a-linha, diretamente do código-fonte
  - Mais lentas, porém muito mais flexíveis e produtivas
- Três construções são usadas em linguagens:
  - Sequência
  - Seleção / Condicionais
  - Repetição / Laços

Compilador

- Compilador: transforma código-fonte em código binária (executável pela CPU)



Exemplo de Algoritmo

- Exemplo: Chegar à um local
  - Em um algoritmo, cada linha é uma instrução a ser executada
- Três construções são usadas:
  - Sequência
  - Seleção / Condicionais
  - Repetição / Laços

```
INÍCIO
    ande 1 quadra
    vire à esquerda 90º
    ande 2 quadras
    vire à direita 45º
    enquanto não chegar ao prédio azul {
        ande 1 quadra
        desvie de obstáculos
    }
    se porta aberta {
        entre
    } senão {
        deixe recado na caixa de correios
    }
FIM
```

Linguagem C/C++

Material e ferramentas

- Padrões
  - Linguagem C ISO: C90, C99, C11
  - Linguagem C++ ISO: C++99, **C++11**, C++14, C++17
- Ótimos tutoriais
  - https://www.tutorialspoint.com/cprogramming/ (C)
  - https://www.tutorialspoint.com/cplusplus/ (C++)
    - ⋆ Nosso material será baseado neste tutorial
  - http://www.cplusplus.com/doc/tutorial/ (C++)
- Possibilitam editar e testar os códigos de exemplo diretamente dentro da página
- Contém definições atualizadas das linguagens C e C++
- Para testes simples: http://cpp.sh/
- Na disciplina: http://www.codeblocks.org/

Primeiro programa

```c
// inclui a biblioteca standard input/output, que contém
// a função printf() usada abaixo
#include <stdio.h>
// em C/C++ todo programa começa pela função main()
int main () {
    // função printf() imprime o conteúdo passado
    // texto (string) deve estar entre aspas
    printf("Welcome to the world of programming!\n");
}
```

- Crie o arquivo soma1.c com o conteúdo acima em um editor de textos
  - $ gcc soma1.c -o soma1.bin (para compilar, gerar programa)
  - $ ./soma1.bin (para executar)
- Um programa é formado por linhas com comandos, que usam:
  - palavras reservadas (usualmente operadores, palavras, símbolos, . . .)
  - funções: main() e printf() são funções
- Questão: Como imprimir várias linhas?

C vs C++

- Programa C

```c
// inclui a biblioteca standard input/output, que contém a função printf()
#include <stdio.h>
// em C todo programa começa pela função main()
int main ()
{ // início de bloco
    // função printf() imprime o conteúdo passado como parâmetro
    // texto (string) deve estar entre aspas
    printf("Welcome to the world of programming!\n");
} // final de bloco
```

- Programa C++

```cpp
// inclui a biblioteca input/ouput streams, que contém cout, um objeto que
// representa a saída para o console (interface em texto)
#include <iostream>
// assim como em C, C++ precisa da função main() para o início do programa
int main() {
    // std (standard) indica que cout está dentro da "pasta"(espaço de nomes) std

    // espaços de nomes (NAMESPACES) são usados para organizar as bibliotecas. Na
    // linha abaixo sabemos que cout pertence à biblioteca C++ padrão (std = standar

    // texto é direcionado ao console pelo operador "<<"
    std::cout << "Hello World!";
}
```

C++

- Se vamos usar várias funções de um NAME SPACE (“pasta”), podemos omitir seu nome
  - std::cout representa um nome absoluto
  - informando no início do programa que utilizarems o NAME SPACE STD

```
#include <iostream>

// informamos que utilizaremos os recursos do espaço de nomes STD
using namespace std;

int main() {
    // Uma nova linha pode ser iniciada por "\n" no texto ou
    // pelo modificador endl (end of line), que informa ao objeto
    // cout o término da linha
    cout << "Hello World!\n";
    cout << "Programming in C++\n" << endl;
    cout << "Has some advantages\n" << endl;
    cout << "NAME SPACE helps developers to organize libraries\n";
}
```

Tipos de Dados

Tipos de Dados

- Tipos mais comuns

Tipo	Descrição	Literal
int	número inteiro	0, -5, 5678
float	número real (com casas decimais)	3.4f, -0.005
char	inteiro que indexa caractere em ASCII	'A', '5', 'b', '#', ')'
string	texto (não é um tipo simples)	"João Sauro", "Res: 15.6"

- Valores literais
  - Valores declarados explicitadamente no código, ao invés de serem calculados pela lógica
- Para o tipo inteiro e similares, podemos aplicar modificadores:
  - unsigned, signed, short, long

Tipos de Dados

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Figure 4

Programa: tipos e tamanhos

```c
// C
#include <stdio.h>
int main() {
    printf("int %d bits\n", sizeof(int)*8);
    printf("long %d bits\n", sizeof(long)*8);
    printf("char %d bits\n", sizeof(char)*8);
    // C não tem tipo explícito boolean...
    printf("\n");
    printf("float %d bits\n", sizeof(float)*8);
    printf("double %d bits\n", sizeof(double)*8);
}
```

```cpp
// C++
#include <iostream>
using namespace std;
int main() {
    cout << "int  " << sizeof(int)*8 << " bits" << endl;
    cout << "long " << sizeof(long)*8 << " bits" << endl;
    cout << "char " << sizeof(char)*8 << " bits" << endl;
    cout << "bool " << sizeof(bool)*8 << " bits" << endl;
    cout << endl;
    cout << "float " << sizeof(float)*8 << " bits" << endl;
    cout << "double " << sizeof(double)*8 << " bits" << endl;
}
```

Variáveis, Constantes e E/S

Variáveis

- **Variável**: espaço de memória para guardar um valor
  - Caracterizada por: Nome, tipo, Tamanho, Alocação e Escopo
- **Tipo**: indica um dos tipos aceitos na linguagem
- **Nome/Identificador**: pode conter letras, números e alguns símbolos. Não pode conter espaço ou iniciar com número.
- **Tamanho**: relativo ao tipo a ser guardado
- **Alocação da memória**: quando declaramos uma variável, o compilador geracódigo para alocar e desalocar variáveis (alocação automática)
  - Existe também a alocação manual de memória, que veremos mais adiante
- **Escopo**: região do código em que pode ser vista. De forma simplificada, uma variável pode ser vista dentro do bloco { } onde foi declarada.

Variáveis: atribuindo literais

```c
#include <stdio.h>

int main() {
    int numero = -5;     // variáveis recebendo valores literais
    int soma;            // qual valor tem a variável soma?
    char letra = 65;
    char letra2 = 'B';
    float real = 45.6f;
    int existe = 1;      // true -> verdadeiro
    char nome[] = "Joao da Silva";

    printf("numero %d\n", numero);
    printf("soma   %d\n", soma); // qual valor imprime?
    printf("letra  %c\n", letra); // qual valor imprime?
    printf("letra2 %c\n", letra2);
    printf("real   %f\n", real);
    printf("existe %d\n", existe);
    printf("nome   %s\n", nome);;
}
```

Variáveis: copiando valores entre variáveis

```c
#include <stdio.h>

int main() {
    int valorA = 35;
    char nomeA[] = "Joao da Silva";

    // variáveis recebendo cópias dos valores de outras variáveis
    int valorB = valorA;
    char nomeB[40];
    strcpy(nomeB, nomeA);
    printf("valorA %d\n", valorA);
    printf("valorB %d\n", valorB);
    printf("nomeA %s\n", nomeA);
    printf("nomeB %s\n\n", nomeB);

    // nomeA e nomeB possem espaços de memória diferentes e não estão ligados, da me
    valorB = 56;
    strcpy(nomeB, "Maria Mercedes");
    printf("valorA %d\n", valorA);
    printf("valorB %d\n", valorB);
    printf("nomeA %s\n", nomeA);
    printf("nomeB %s\n", nomeB);
}
```

Tipos e Casting

- Usamos casting para conversão entre tipos numéricos

```c
#include <stdio.h>

main() {
   double a = 21.09399;
   float b = 10.20;
   int c ;

   c = (int) a;
   printf("Line 1 - Value of (int)a is: %d\n", c);

   c = (int) b;
   printf("Line 2 - Value of (int)b is: %d\n", c);

   return 0;
}
```

Entrada e Saída E/S .. Input and Output I/O

```c
#include <stdio.h>

int main() {
    int numero;
    char frase1[50];
    char frase2[50];

    // ALTAMENTE NÃO RECOMENDADO
    // funcão gets() está marcada como depreciada
    // problema? não impede a leitura e mais de 50 caracteres
    printf("Digite uma frase: \n");
    gets(frase1);

    printf("Digite outra frase: \n");
    fgets(frase2, sizeof(frase2), stdin);

    printf("Digite um número: \n");
    scanf("%d", &numero);

    printf("numero:  %d\n", numero);
    printf("frase1: %s\n", frase1);
    printf("frase2: %s\n", frase2);
}
```

Constantes: #define

- Define uma MACRO, uma diretiva de substituição de valores em tempo de compilação.

```c
#include <stdio.h>

#define LENGTH 10
#define WIDTH  5
#define NEWLINE '\n'

int main() {
   int area;

   area = LENGTH * WIDTH;
   printf("value of area : %d", area);
   printf("%c", NEWLINE);

   // por via de regra, devemos informar ao SO que tudo terminou co
   return 0;
}
```

Constantes: const

- Define uma variável somente leitura (read-only).

```c
#include <stdio.h>

int main() {
    const int  LENGTH = 10;
    const int  WIDTH = 5;
    const char NEWLINE = '\n';
    int area;

    area = LENGTH * WIDTH;
    printf("value of area : %d", area);
    printf("%c", NEWLINE);

    return 0;
}
```