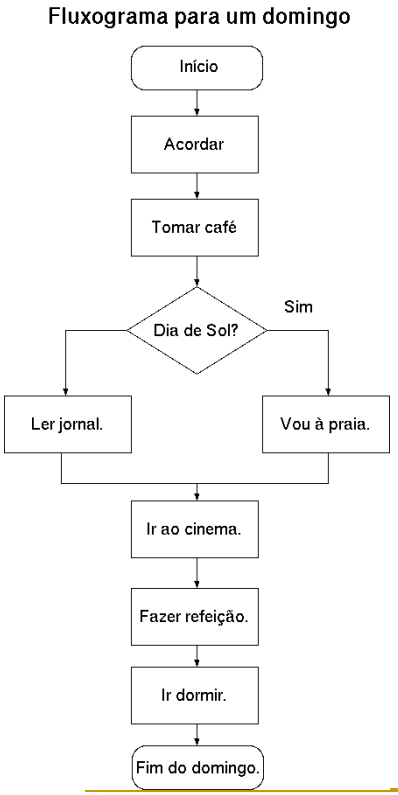
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio Preto



Ciência da Computação

Prof. Dr. Leandro Alves Neves

Aula 04

Algoritmos e Programação

Sumário

- Estruturas de Dados
  - Tipos de Dados
- Identificadores
- Constantes e Variáveis
  - Modelo esquemático
  - Características e Regras para Declaração
  - Atribuição
- Estrutura Básica de um Algoritmo
- Entrada e Saída de Dados: Comandos

Estrutura Básica de Um Algoritmo

- Um algoritmo e/ou programa deve respeitar uma estrutura básica.

Pseudocódigo ou português estruturado

```
programa  <nome>

início
    tipo de dado <nome >;
     <comandos>;
fim
```

Entrada e Saída de Dados: Comandos

- Existem comandos que permitem entradas e saídas de dados ou informações.
  - **programa** nome_idade

```
início
   string nome[40];
   inteiro idade;
   escreva ("Digite seu nome");
   leia (nome);
   escreva ("Digite sua idade");
   leia (idade);
   escreva ("Olá", nome);
   escreva (", sua idade é", idade);
fim
```

Algoritmo versus Programa em C

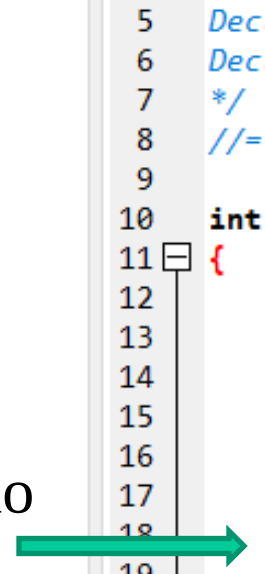
```
programa  nome_idade

início

   string nome[40];
   inteiro idade;
   escreva ("Digite seu nome");
   leia (nome);
   escreva ("Digite sua idade");
   leia (idade);
   escreva ("Olá", nome);
   escreva (", sua idade é",
   idade);
fim
```

Arquivo nome_idade.c

```
#include <stdio.h>
int main ()
{
   char nome[40];
   int idade;
   printf("\nDigite seu nome ");
   scanf("%s",nome);
   printf("\nDigite sua idade ");
   scanf("%d",&idade);
   printf("\nOlá %s",nome);
   printf(", sua idade é %d", idade);
   //system("PAUSE");
   return 0;
}
```

Estrutura de um Programa em C: Exemplo

Indentação



Exemplo_Aula4_slide15.cpp

```
1  //Diretivas de Pré-processamento (Obrigatórias)
2  #include <stdio.h>
3  //=-=-=-=-=-=-=-=-
4  /*Opcional.
5  Declarações de Variáveis Globais, por exemplo.
6  Declarações de Funções. Importante, não vamos estudar em ATPI
7  */
8  //=-=-=-=-=-=-=-=-=-=
9
10 int main ()
11 {   //Obrigatório. Função principal: indica o início do programa
12
13     //=-=-=-=-=-=-=-=-=-=-=-=-
14     //Opcional. Declaração de Variáveis Locais. Estudado em ATPI
15     char nome[40];
16     int idade;
17     //=-=-=-=
18     //Obrigatório. Comandos para resolução do problema
19     printf("\nDigite o seu nome: ");
20     scanf("%s",nome);
21     printf("\nDigite sua idade: ");
22     scanf("%d",&idade);
23     printf("\nOlá %s", nome);
24     printf(" sua idade é: %d", idade);
25
26     //Opcional. Comando para interromper momentaneamente o programa
27     //system("PAUSE");
28     //Retorno ao SO o status do programa
29     return 0;
30 }//Indica o final do programa.
```

Estruturas de Dados

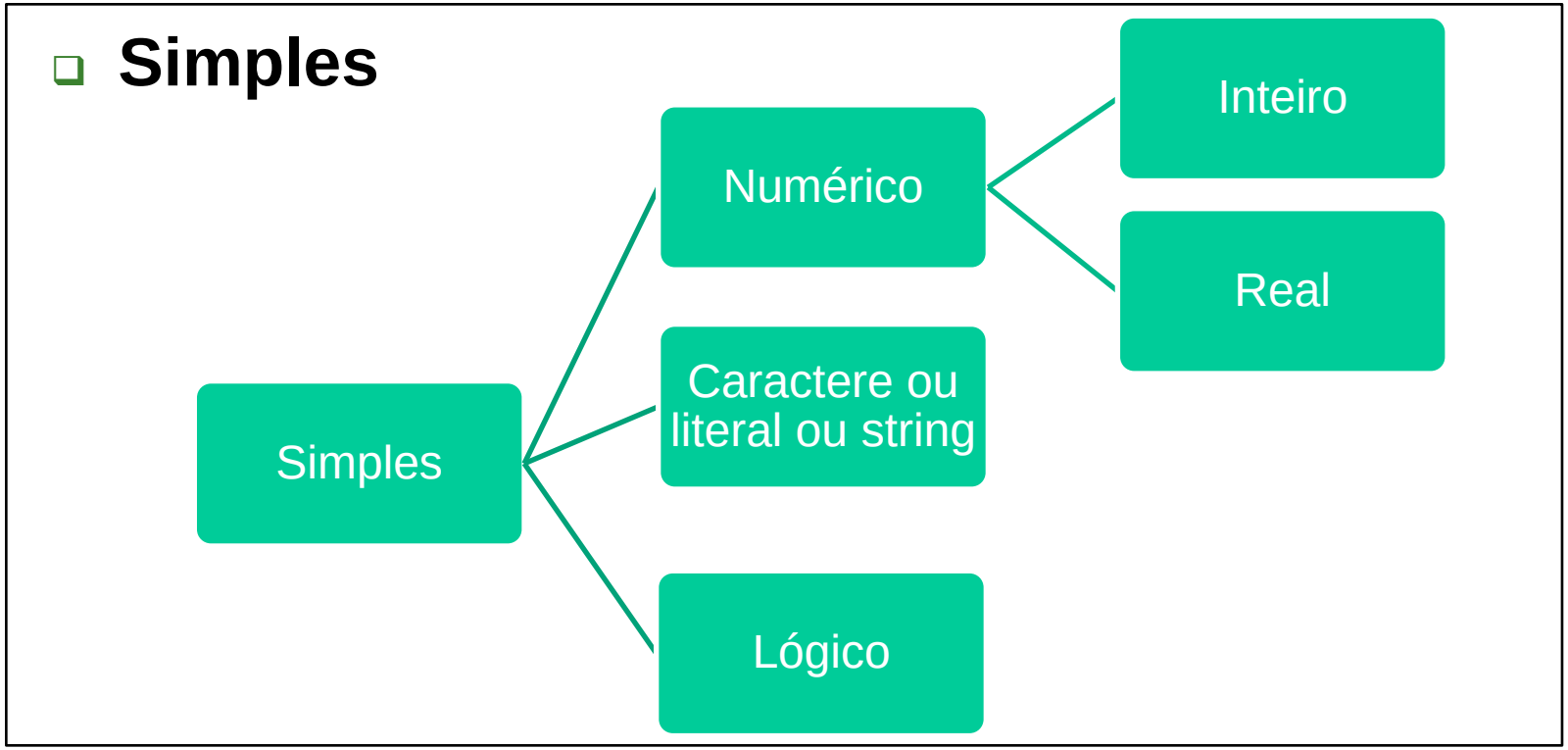
- **Tipos de Dados:**
  - **Categoria de valores** e
  - **Operações**
    - Exemplo: **Conjunto dos inteiros com as operações**:
      - Adição, Subtração, Divisão e Multiplicação

Estruturas de Dados

- **Tipos de Dados:**
  - **Em um algoritmo/programa deve-se constar:**
    - Tipo de dado que será armazenado e/ou manipulado
    - Consequentemente:
      - Quanto espaço de memória é necessário
      - Operações permitidas/existentes

Estruturas de Dados

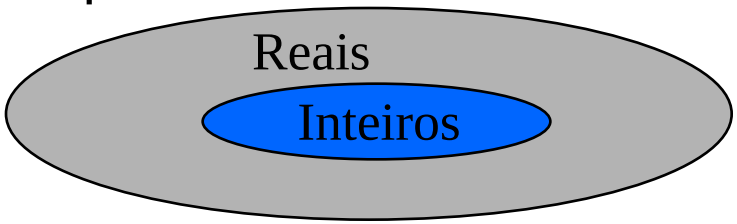
- **Tipos de Dados Básicos:**
  - **Simples**



  - **Composto (vetor, matriz, registros)**

Estruturas de Dados

- **Tipos de Dados Básicos:**
  - **Tipo Inteiro**
    - Positivo, negativo ou nulo ⟹ -23, 98, -357, 0
    - Não tem um componente decimal
  - **Tipo Real**
    - Positivo, negativo ou nulo ⟹ -23.45, 98.1346, -357.71, 0.0
    - Tem um componente decimal



Tipo inteiro ocupa menos espaço de armazenamento em memória do que o **tipo Real**

Estruturas de Dados

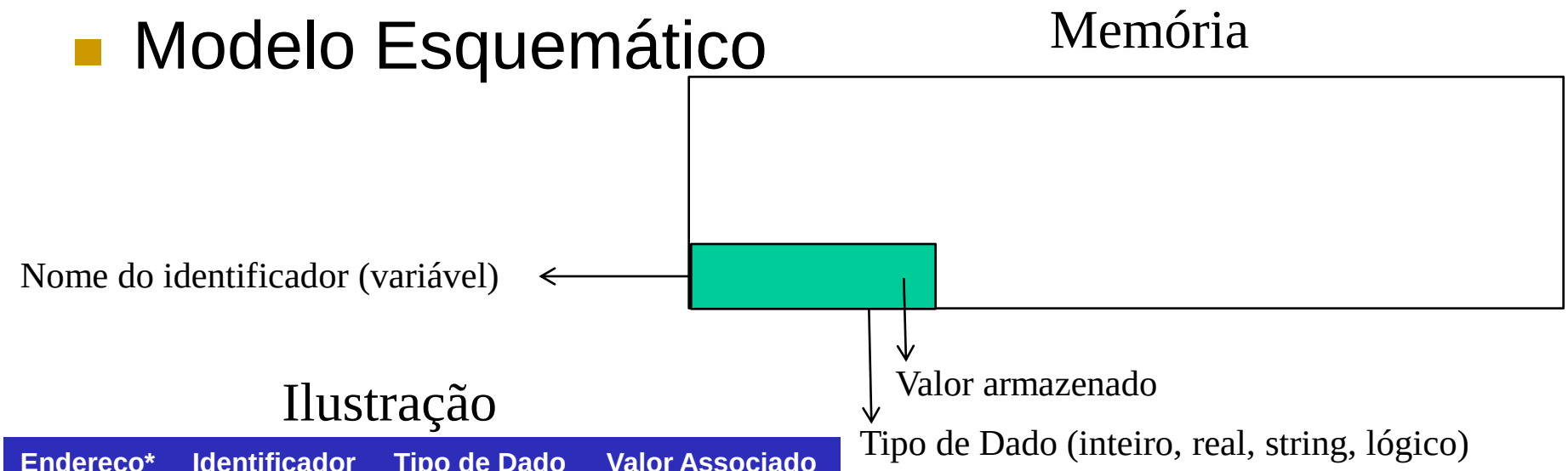
- **Tipos de Dados Básicos:**
  - **Caractere, string ou literal**
    - Alfanuméricos: ***A*** a ***Z*****, *a*** a ***z*****, 0** a **9**, espaço em branco ( ) e símbolos especiais (~, ., ?, >, <, etc).
    - Deve aparecer entre aspas “ ” ou ‘ ’, exemplos:
      - “a”; “algoritmos”; “sala 5c”; “7,2”, etc
      - ‘a’; ‘algoritmos’; ‘sala 5c’; ‘7,2’, etc
  - **Tipo Lógico**
    - Representa o conceito lógico de verdade ou falsidade
    - Utilizados em controle do fluxo lógico do algoritmo
    - Exemplos:
      - **0** ou **1**; **V** ou **F**; **Sim** ou **Não**; **True** ou **False**.

Estruturas de Dados: Variável

- Como armazenar, referenciar ou recuperar dados em um computador?
  - Dados são comumente **armazenados na memória**
  - **Cada tipo de dado** ocupa um espaço (em bytes)
  - **Referenciar ou recuperar** requer o **endereço do byte inicial** na memória
    - Endereço inicial é um identificador
    - Variável é um sinônimo do termo identificador

Variáveis

- Modelo Esquemático



Ilustração

Endereço*	Identificador	Tipo de Dado	Valor Associado
2001	nome	caractere	“Maria”
2002	numero	inteiro	4356
2003	letra	caractere	“A”
2004	resposta	lógico	True
2005	peso	real	75.4

*Valores ilustrativos para fins didáticos

Variáveis

- Características:
  - Valores podem ser alterados durante o programa
    - Porém, um único valor por vez
  - Valores armazenados devem ser do mesmo tipo de dado daquele para o qual a variável foi criada

Variáveis

- **Regras para definir nomes válidos**:

1. Usar apenas letras e números;
2. O primeiro caractere deve ser uma letra
3. Caracteres especiais (inclusive espaço em branco) não são válidos
   - Exceto: “_”

- Exemplos:
  - Inválidos: @A 2B F 1 índice salário mês valor 1
  - Válidos: F_1 indice salario_mes valor_1 a

Estrutura de um Programa em C: Variáveis

Declaração de Variável local

Exemplo_Aula4_slide15.cpp

```
1   //Diretivas de Pré-processamento (Obrigatórias)
2   #include <stdio.h>
3   //=-=-=-=-=-=-=-=-
4   /*Opcional.
5   Declarações de Variáveis Globais, por exemplo.
6   Declarações de Funções. Importante, não vamos estudar em ATPI
7   */
8   //=-=-=-=-=-=-=-=-=-=
9
10  int main ()
11  {   //Obrigatório. Função principal: indica o início do programa
12
13      //=-=-=-=-=-=-=-=-=-=-=-=-
14      //Opcional. Declaração de Variáveis Locais. Estudado em ATPI
15      char nome[40];
16      int idade;
17      //=-=-=-=
18      //Obrigatório. Comandos para resolução do problema
19      printf("\nDigite o seu nome: ");
20      scanf("%s",nome);
21      printf("\nDigite sua idade: ");
22      scanf("%d",&idade);
23      printf("\nOlá %s", nome);
24      printf(" sua idade é: %d", idade);
25
26      //Opcional. Comando para interromper momentaneamente o programa
27      //system("PAUSE");
28      //Retorno ao SO o status do programa
29      return 0;
30  }//Indica o final do programa.
```

Tipos de Dados em Linguagem C

- **Há cinco tipos básicos de dados**
  - Caractere: **char**
  - Inteiro: **int**
  - Ponto flutuante: **float**
  - Ponto flutuante de precisão dupla: **double**
  - Sem valor: **void**

Tipos de Dados em Linguagem C

- **Lembrete, complemento de 2**

Decimal	Binário s/ sinal	Binário (Compl. 2)
-8	-	1000
-7	-	1001
-6	-	1010
-5	-	1011
-4	-	1100
-3	-	1101
-2	-	1110
-1	-	1111
0	000	0000
1	001	0001
2	010	0010
3	011	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111

Usando 4 bits

Sinal, bit mais significativo
0: positivo
1: negativo

Leitura Representação Complemento de 2: material já disponibilizado

Tipos de Dados em Linguagem C

- **Tipos básicos podem ser modificados**
  - **Números naturais** (0, 1, 2, 3, …): **inteiros sem sinal**
    - Implementado por meio do modificador **unsigned**
    - **unsigned int i;**
  - **Números inteiros** (…, −2, −1, 0, +1, +2, …): **inteiros com sinal.**
    - **int i;**

Leitura Representação Complemento de 2: material já disponibilizado

Tipos de Dados em Linguagem C

- **Tamanho e faixa de dados variam**
  - **De acordo com a geração do processador e compilador, por exemplo**
    - Padrão ANSI (American National Standards Institute)
    - Define apenas a **faixa mínima** de cada tipo

Tipo	Tamanho aproximado em bits	Faixa mínima
char	8	-127 a 127
unsigned char	8	0 a 255
signed char	8	-127 a 127
int	16	-32.767 a 32.767
unsigned int	16	0 a 65.535
signed int	16	O mesmo que **int**
long int	32	-2.147.483.647 a 2.147.483.647
signed long int	32	O mesmo que **long int**.
unsigned long int	32	0 a 4.294.967.295
float	32	Seis dígitos de precisão
double	64	Dez dígitos de precisão
long double	80	Dez dígitos de precisão

Estrutura de um Programa em C: Identificadores

Identificadores

Exemplo_Aula4_slide15.cpp

```
1  //Diretivas de Pré-processamento (Obrigatórias)
2  #include <stdio.h>
3  //=-=-=-=-=-=-=-=-
4  /*Opcional.
5  Declarações de Variáveis Globais, por exemplo.
6  Declarações de Funções. Importante, não vamos estudar em ATPI
7  */
8  //=-=-=-=-=-=-=-=-=-=
9
10 int main ()
11 {   //Obrigatório. Função principal: indica o início do programa
12
13     //=-=-=-=-=-=-=-=-=-=-=-=-
14     //Opcional. Declaração de Variáveis Locais. Estudado em ATPI
15     char nome[40];
16     int idade;
17     //=-=-=-=
18     //Obrigatório. Comandos para resolução do problema
19     printf("\nDigite o seu nome: ");
20     scanf("%s",nome);
21     printf("\nDigite sua idade: ");
22     scanf("%d",&idade);
23     printf("\nOlá %s", nome);
24     printf(" sua idade é: %d", idade);
25
26     //Opcional. Comando para interromper momentaneamente o programa
27     //system("PAUSE");
28     //Retorno ao SO o status do programa
29     return 0;
30 }//Indica o final do programa.
```

Estrutura de um Programa em C: Identificadores

- **Printf()**

Código	Formato
%c ⟷	Caractere
%d ⟷	Inteiros decimais com sinal
%i	Inteiros decimais com sinal
%e	Notação científica (e minúsculo)
%E	Notação científica (E maiúsculo)
%f ⟷	Ponto flutuante decimal
%o	Octal sem sinal
%s ⟷	String de caracteres
%u	Inteiros decimais sem sinal
%x	Hexadecimal sem sinal (letras minúsculas)
%X	Hexadecimal sem sinal (letras maiúsculas)
%p ⟷	Apresenta um ponteiro
%lf	Apresenta um tipo double

Permite apresentar valores na base 8, 10 e 16.

⟷ Frequentemente utilizados

Estrutura de um Programa em C: Identificadores

- **Scanf()**

Código	Significado
%c ⟷	Lê um único caractere
%d ⟷	Lê um inteiro decimal
%i	Lê um inteiro decimal →
%f ⟷	Lê um número em ponto flutuante
%o	Lê um número octal
%s ⟷	Lê uma string
%x	Lê um número hexadecimal
%p ⟷	Lê um ponteiro
%u	Lê um inteiro sem sinal
%lf	Lê um tipo double

Permite entradas de valores na base 8, 10 e 16.

⟷ Frequentemente utilizados

Estrutura de um Programa em C: Identificadores

```c
#include <stdio.h>

int main()
{  float teste;

   printf(" --- TIPO ---|--- BYTES ---\n");
   printf(" char ..............: %lu bytes\n", sizeof(char));
   printf(" unsigned char.....: %lu bytes\n", sizeof(unsigned char));
   printf(" int................: %lu bytes\n", sizeof(int));
   printf(" unsigned int......: %lu bytes\n", sizeof(unsigned int));
   printf(" long int..........: %lu bytes\n", sizeof(long int));
   printf(" unsigned long int.: %lu bytes\n", sizeof(unsigned long int));

   printf(" long float .......: %lu bytes\n", sizeof(float));
   printf(" double............: %lu bytes\n", sizeof(double));
   printf(" long double.......: %lu bytes\n\n", sizeof(long double));
   printf("\nO tamanho de teste.............: %lu \n\n",sizeof teste);

   return 0;
}
```

```
> clang-7 -pthread -lm -o main main.c
> ./main
 --- TIPO ---|--- BYTES ---
 char ..............: 1 bytes
 unsigned char.....: 1 bytes
 int................: 4 bytes
 unsigned int......: 4 bytes
 long int..........: 8 bytes
 unsigned long int.: 8 bytes
 long float .......: 4 bytes
 double............: 8 bytes
 long double.......: 16 bytes


O tamanho de teste.............: 4

>
```

- **Exemplos**

Estrutura de um Programa em C: Identificadores

```c
#include <stdio.h>

int main()
{  float teste;

   printf(" --- TIPO ---|--- BYTES ---\n");
   printf(" char ..............: %lu bytes\n",
sizeof(char));
   printf(" unsigned char.....: %lu bytes\n",
sizeof(unsigned char));
   printf(" int................: %lu bytes\n", sizeof(int));
   printf(" unsigned int......: %lu bytes\n", sizeof(
unsigned int));
   printf(" long int..........: %lu bytes\n", sizeof(long
int));
   printf(" unsigned long int.: %lu bytes\n",
sizeof(unsigned long int));

   printf(" long float .......: %lu bytes\n", sizeof(
float));
   printf(" double............: %lu bytes\n",
sizeof(double));
   printf(" long double.......: %lu bytes\n\n", sizeof(long
 double));
   printf("\nO tamanho de teste.............: %lu
\n\n",sizeof teste);

   return 0;
}
```

```
:15:00 UTC 2022 x86_64 GNU/Linux
> gcc --version
gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is
 NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURP
OSE.

> gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.5.0-3u
buntu1~18.04' --with-bugurl=file:///usr/share/doc/gcc-7/README.Bugs --e
nable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++ --prefix=/usr
--with-gcc-major-version-only --program-suffix=-7 --program-prefix=x86_
64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/us
r/lib --without-included-gettext --enable-threads=posix --libdir=/usr/l
ib --enable-nls --enable-bootstrap --enable-clocale=gnu --enable-libstd
cxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new
--enable-gnu-unique-object --disable-vtable-verify --enable-libmpx --en
able-plugin --enable-default-pie --with-system-zlib --with-target-syste
m-zlib --enable-objc-gc=auto --enable-multiarch --disable-werror --with
-arch-32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable
-multilib --with-tune=generic --enable-offload-targets=nvptx-none --wit
hout-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --h
ost=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)
>
```

A&P

Estrutura de um Programa em C: Identificadores

Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda

TDM-GCC 4.9.2 32-bit Profiling

(globals)

Depura Exemplo_2.cpp

```
1  #include <stdio.h>
2
3  int main()
4  { float teste;
5
6    printf(" --- TIPO ---|--- BYTES ---\n");
7    printf(" char ..............: %lu bytes\n
8    printf(" unsigned char.....: %lu bytes\n
9    printf(" int................: %lu bytes\n
10   printf(" unsigned int......: %lu bytes\n
11   printf(" long int..........: %lu bytes\n
12   printf(" unsigned long int.: %lu bytes\n
13
14   printf(" long float .......: %lu bytes\n
15   printf(" double............: %lu bytes\n
16   printf(" long double.......: %lu bytes\n
17   printf("\nO tamanho de teste.....
18
19   return 0;
20 }
```

E:\New_data\Recovered data 04-21 11_40_43\Resultado de Digitaliza...

```
--- TIPO ---|--- BYTES ---
char ..............: 1 bytes
unsigned char.....: 1 bytes
int................: 4 bytes
unsigned int......: 4 bytes
long int..........: 4 bytes
unsigned long int.: 4 bytes
long float .......: 4 bytes
double............: 8 bytes
long double.......: 16 bytes


O tamanho de teste.....: 4


--------------------------------
Process exited after 0.5749 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Variáveis: Atribuição

- Processo que permite associar valores a uma variável:

Atribuições Válidas (Linguagem C)

```
float salario_mes;
int idade;
.....
salario_mes = 28.00;
idade = 25;
```

Errado!

```
float salario_mes;
int idade;
.....
salario_mes = “vinte e oito”;
idade = “vinte e cinco;
```

Constantes: Const e #define

- Constantes: não podem ser modificadas pelo Programa
  - Exemplos:
  - a = **999**;
  - c = **'a'**;
  - pi=3.1415;

  - Tipos:
    - **const**
    - **#define**

Constantes: Const e #define

- **const**
  - Declaração, exemplo:
    - const int a=999;
    - const float pi=3.1415;

```c
#include <stdio.h>

int main ()
{

    const char a='l';
    const float pi=3.1415;
    const int n= -1;

    printf("\n%i", n);
    printf("\n%.4f", pi);
    printf("\n%c", a);

    printf("\nExemplo de operação: %.4f", pi * n);

    return 0;
}
```

  - **Protege os objetos apontados pelos argumentos em uma função**

Constantes: Const e #define

- **#define**
  - É uma macro
  - Realiza a associação de um identificador com um valor

```c
#include <stdio.h>

#define a 'l'
#define pi 3.1415
#define n -1

int main ()
{

    printf("\n%i", n);
    printf("\n%.4f", pi);
    printf("\n%c", a);

    printf("\nExemplo de operação: %.4f", pi * n);

    return 0;
}
```

  - **Compilador substitui as ocorrências (identificador) pelo valor**
    - **Etapa de pré-processamento**

  - **Mais eficiente**: não usa memória

Constantes: Const e #define

- **Visualização**

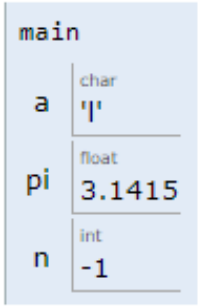
C (gcc 9.3, C17 + GNU extensions)
(known limitations)

```
1  #include <stdio.h>
2  #define pi2 3.141592
3  int main ()
4  {
5
6      const char a='l';
7      const float pi=3.1415;
8      const int n= -1;
9
10     printf("\n%i", n);
11     printf("\n%.4f", pi);
12     printf("\n%.6f", pi2);
13     printf("\n%c", a);
14
15     printf("\nExemplo de operação: %.4f", pi * n);
16
17     return 0;
18 }
```

Print output (drag lower right corner to resize)

```
-1
3.1415
3.141592
l
Exemplo de operação: -3.1415
```

Stack Heap



http://www.pythontutor.com/visualize.html#mode=display

Até aqui vimos o seguinte:

- Estruturas de Dados
- Tipos de Dados
- Identificadores
- Constantes e Variáveis
- Estrutura Básica de um Algoritmo
- Entrada e Saída de Dados: Comandos

- Próximo Conteúdo:
  - Operadores e Expressões (Relacionais, Aritméticas e Lógicas)

Bibliografia Complementar

- SCHILDT, H. C Completo e Total, 3ª ed., Pearson 1996. 852p.
  - Capítulo 8: Entrada e saída, identificadores e outros,198 a 218
  - Capítulo 12, especialmente as páginas 314 a 316 (printf) e 322 a 326 (scanf).
  - Páginas 9 a 13: A Forma de Um Programa em C
  - Páginas 16 a 19: Tipos Básicos de Dados, Modificando os Tipos Básicos e Identificadores
  - Páginas 20 a 24: Variáveis e Variáveis Locais
  - Página 27: Modificadores de tipo de acesso (Const)
  - Página 255 a 258: #define e #include
  - Página 36 e 37: Inicialização de Variáveis
  - Página 38: Constantes Caractere de Barra Invertida
  - Páginas 39 e 40: Operador de Atribuição
  - Páginas 266 e 267: Comentários

Bibliografia Complementar

- SALES, André Barros de; AMVAME-NZE, Georges Daniel. Linguagem C: roteiro de experimentos para aulas práticas [recurso eletrônico]. Florianópolis: UFSC, 2016. Disponível em: <http://repositorio.unb.br/handle/10482/21540>.
  - Leitura (Obrigatório):
    - Capítulo 1 (página 6 a 11);
    - Capítulo 2 (14 a 22);
  - Realizar os Experimentos (adaptado para https://replit.com/languages/c)
  - Realizar as Atividades de Fixação (Obrigatório)
    - Páginas: 10 e 11; 20 e 21