Engenharia da Computação

www.eComp.Poli.br

Tipos de Dados Especiais em C

Disciplina: **DCExt Programação Imperativa**

Prof. Hemir Santiago

hcs2@poli.br

Material cedido pelo Prof. Joabe Jesus

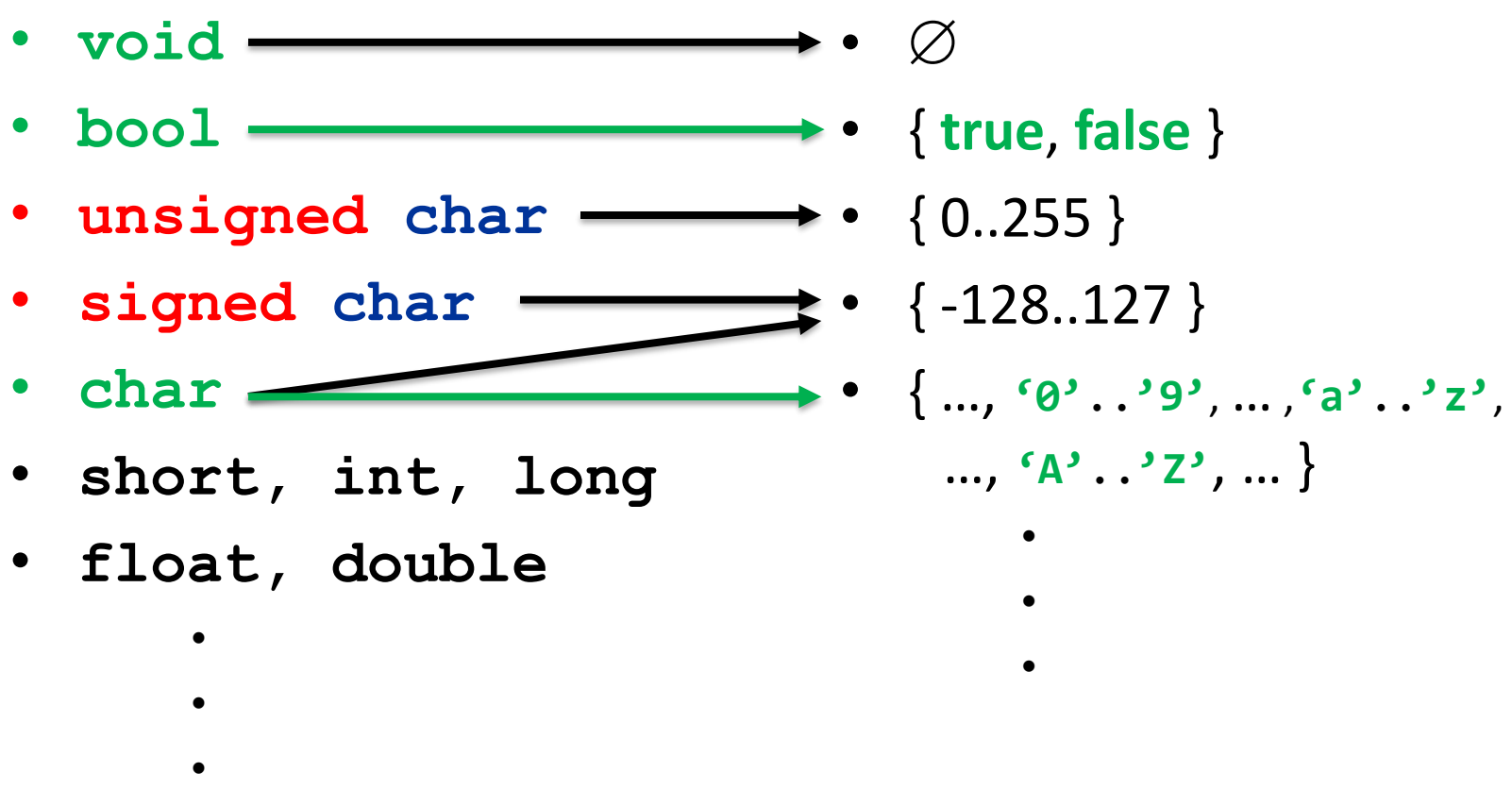
Objetivos

- Discutir os conceitos de Tipos de Dados Especiais como:
  - booleano,
  - void,
  - caracteres,
  - vetores,
  - matrizes e
  - *strings*

Tipos de Dados em C

Tipo de Dados em C	Conjunto de Valores do Tipo
• `unsigned char`	• { 0..255 }
• `signed char`	• { -128..127 }
• `char`	
• `short, int, long`	
• `float, double`	
•	•
•	•
•	•

Tipos de Dados Especiais



Tipos de Dados Especiais

- **`void`** (Tipo vazio / Sem Tipo)
  - Não possui valores
  - Quando uma função C descreve um procedimento podemos usar o tipo **vazio** como tipo de retorno
    - indica que **não possui valor** de retorno

Tipos de Dados Especiais

- **Booleano** (valor de predicado/lógica)
  - Pode ser utilizada qualquer variável inteira
  - O programa identificará o valor lógico/booleano como False (Falso) apenas se o valor for igual a 0

    0 → Falso

    ≠ 0 → Verdadeiro
  - Variáveis booleanas são usadas principalmente como o resultado de operadores relacionais

    ==, !=, >, <, >=, <=

Tipos de Dados Especiais

- **Booleano**
  - Também podemos utilizar o tipo **`bool`**
    - Disponível na biblioteca `<stdbool.h>`
    - Permite os valores **`true`** e **`false`**
  - **Exemplo**:

```
#include <stdbool.h>
...
int a = 1;
bool b = false;
b = a; // vai fazer b ser true
```

Tipos de Dados Especiais

- **Caracteres**
  - Para criar variáveis que armazenam caracteres, podemos usar o tipo **char**
    - Internamente (na memória) será armazenado um valor inteiro representando o código de um caractere **ASCII**
  - A tabela ASCII apresenta um conjunto de 128 sinais: 95 sinais gráficos (letras do alfabeto latino, algarismos arábicos, sinais de pontuação e sinais matemáticos) e 33 sinais de controle.

Códigos ASCII (*ASCII Codes*)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

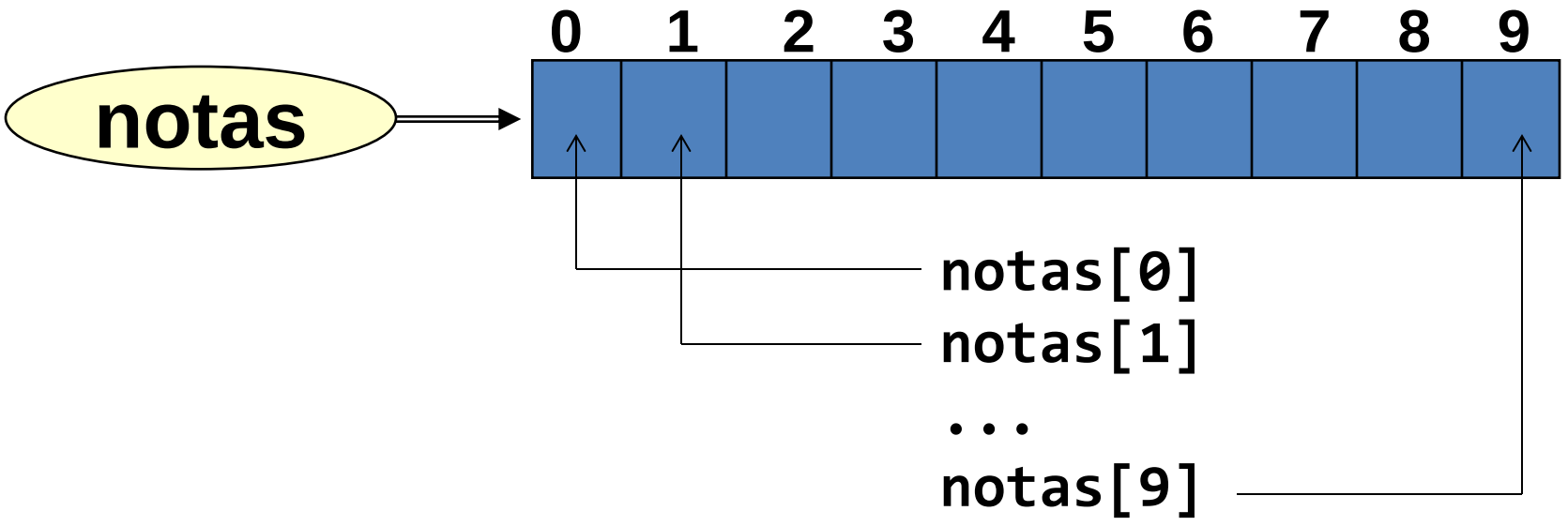
Vetores

- O que acontece se você desejar encontrar a média dos 200 alunos de uma escola?
- Solução: usar um vetor com 200 itens
  - Um vetor é um tipo de dado utilizado para representar um conjunto de valores homogêneos utilizando um único nome.
  - Cada valor é diferenciado através do índice do vetor
  - Em C, o primeiro elemento tem índice 0.

Vetores

```
int notas[10];
```

- **Aloca 10 valores inteiros referenciados através do identificador `notas`.**



Exemplo: média de notas

SEM *ARRAYS*

- Observe:

```
int nota1, nota2, nota3;
printf("Digite a nota do aluno 1: ");
scanf("%d",&nota1);
printf("Digite a nota do aluno 2: ");
scanf("%d",&nota2);
printf("Digite a nota do aluno 3: ");
scanf("%d",&nota3);
printf("A média dos alunos é: %.2f",
                    (nota1+nota2+nota3)/3.0);
```

Exemplo: média de notas

COM *ARRAYS*

```
int notas[3];
int soma;

printf("Digite a nota do aluno 1: ");
scanf("%d", &notas[0]);
printf("Digite a nota do aluno 2: ");
scanf("%d", &notas[1]);
printf("Digite a nota do aluno 3: ");
scanf("%d", &notas[2]);

soma = notas[0] + notas[1] + notas[2];
printf("Média = %.2f\n“, soma/3.0);
```

Exemplo: média de notas

COM *ARRAYS E REPETIÇÕES*

```
int notas[3];
int soma, i;

for (i = 0; i < 3; i++) {
   printf("Digite a nota do aluno %d: ",i);
   scanf("%d",&notas[i]);
}
soma = 0;
for (i = 0; i < 3; i++)
   soma = soma + notas[i];
printf("Média = %.2f\n“,soma/3.0);
```

Exemplo: média de notas

COM *ARRAYS, REPETIÇÕES E FUNÇÕES*

```
int notas[3];
void lerNotas() {
     int i;
     for (i = 0; i < 3; i++) {
       printf("Digite a nota do aluno %d: ", i);
       scanf("%d", &notas[i]);
     }
}
int somaNotas() {
     int soma = 0, i;
     for (i = 0; i < 3; i++)
       soma = soma + notas[i];
    return soma;
}
void main() {
     int soma;
     lerNotas();
     soma = somaNotas();
     printf("Média = %.2f\n", soma/3.0);
}
```

Inicializando Vetores

```
int tab[5] = {7, 0, -9, 15, 38};
int fib[] = {1,1,2,3,5,8,13,21,34};
```

- **Na ausência do tamanho do vetor, o compilador contará o total de itens na lista de inicialização**
- **Se o programador declarar um vetor sem inicializá-lo, ele deverá declarar sua dimensão**
- **Se fizer os dois**
  - **Se há menos inicializadores que a dimensão especificada, os outros serão zero**
  - **Mais inicializadores que o necessário implica em *warning***

Cuidados

- C não avisa quando o limite de um vetor é excedido!

- O programador tem a responsabilidade de verificar o limite do vetor

Matrizes

- Aprendemos a declarar vetores

```
int notas[10];
```

- Mas podemos usar vetores de mais de uma dimensão –> MATRIZES

```
int matriz1[3][3]; ([linhas][colunas])
int matriz2[][2]={{1,2},{5,6}};
int matriz3[2][2]={0,1,2,3};
```

Percorrendo matrizes (**SEM** uso de **Repetições**)

```
int matriz[2][2]={{1,2},{5,6}};

printf("%d ", matriz[0][0]);
printf("%d\n",matriz[0][1]);
printf("%d ", matriz[1][0]);
printf("%d\n",matriz[1][1]);
```

Percorrendo matrizes

COM REPETIÇÕES

```
int i,j;
int matriz[2][2]={{1,2},{5,6}};
for (i=0;i<2;i++)
{
  for (j=0;j<2;j++)
  {
     printf("%d ",matriz[i][j]);
  }
  printf("\n");
}
```

Exercício #1

- Crie um programa que leia um vetor de 10 números inteiros e que calcule e imprima, depois de ler todos os números:
  - A soma dos números
  - A multiplicação dos números nas posições pares
  - A média ponderada dos números (peso = posição do número no vetor)

Exercício #2

- Fazer um programa que pede uma string ao usuário
- Imprimir na tela os caracteres da string juntamente com seus códigos ASCII, um abaixo do outro

Strings

- Podemos armazenar uma sequência de caracteres (string) em um vetor:

```
char nome[5];
```

  - Como C não controla automaticamente o limite do vetor, devemos sinalizar o final do string com o caractere especial `'\0'`

```
nome[0] = 'c';  nome[1] = 'a';
nome[2] = 's';  nome[3] = 'a';
nome[4] = '\0';
```

Strings constantes

```
printf("Um string constante!\n");
printf("%s fica muito longe","Plutão");
```

ERRADO

```
char nome[10] = {"Corrida"};
nome = “Corrida”;
```

CORRETO

```
char nome[10] = "Corrida"; OU
char nome[10] =
  {'C','o','r','r','i','d','a','\0'};
```

Funções para Strings

- Definidas em `string.h`:

- char str1[5]; char str2[5];
  - `strcpy(str1, str2); //copia str2 em str1`
  - `strcat(str1, str2); //concatena str1 e str2`
  - `strcmp(str1, str2); //retorna 1 se as duas são idênticas e 0 se são diferentes`

Funções para Strings

- Definidas em `string.h`:
  - `strlen(str1); //retorna o tamanho da string`
  - `sprintf(str1, "Valor de Pi = %f", M_PI);`
  - `gets(str1); //lê a string digitada pelo usuário`
  - `puts(str1); //exibe a string`

Exemplo

```
char nome[21];
int ano[2];
printf("Qual eh seu nome? ");
gets(nome);
printf("%s, em que ano estamos? ", nome);
scanf("%d", &ano[0]);
printf("%s, em ano voce nasceu? ", nome);
scanf("%d",&ano[1]);
printf("%s, sua idade eh %d anos.\n", nome, ano[0]-ano[1]);
```

Exemplo – fazer para ver o resultado

```
char msg[81], nome[21], sobrenome[21];
int idade;
printf("Qual é seu nome? ");
gets(nome);
printf("Qual é seu sobrenome? ");
gets(sobrenome);
printf("Qual é sua idade? ");
scanf("%d", &idade);
strcpy(msg,nome);
strcat(msg," ");
strcat(msg,sobrenome);
sprintf(msg,"%s tem %d anos de idade",msg,idade);
puts(msg);
```

Exercício #3

- Elaborar um programa que declara dois arrays: *char nomeAluno[]* e *nomeDisciplina[]*. Em seguida, o programa deve solicitar ao usuário que digite o seu nome e o da disciplina que ele cursa. Para tanto, sugere-se o uso da função *gets*, exibindo o que foi digitado pelo usuário no formato usando a função *sprintf*:

"Bom dia nomeAluno

Seja bem-vindo ao curso de nomeDisciplina"

	DATA	AULA
1	22/08/2024	Apresentação da disciplina | Introdução à Programação Imperativa
2	29/08/2024	Introdução à Linguagem de Programação C
3	05/09/2024	Conceitos Fundamentais
4	12/09/2024	Tipos de Dados Especiais em C
5	19/09/2024	Estruturas Condicionais e de Repetição
6	26/09/2024	Pré-processamento
7	03/10/2024	Registros e Estruturas de Dados
8	10/10/2024	Ponteiros
9	17/10/2024	1º Exercício Escolar

Plano de Aulas

	DATA	AULA
10	24/10/2024	Arquivos
11	31/10/2024	Acompanhamento de projetos
12	07/11/2024	Acompanhamento de projetos
13	14/11/2024	Acompanhamento de projetos
14	21/11/2024	Acompanhamento de projetos
15	28/11/2024	Apresentação parcial
16	05/12/2024	Apresentação de projetos
17	12/12/2024	Avaliação Final

Plano de Aulas