Aula 4:
Caracteres e Números Aleatórios

***Disciplina*: Fundamentos de Programação**

Prof. Luiz Olmes

olmes@unifei.edu.br

UNIFEI
UNIVERSIDADE FEDERAL DE ITAJUBÁ

Nas aulas anteriores...

- **O QUE JÁ ESTUDAMOS?**
  - Algoritmos.
  - Linguagem C.
  - Variáveis e Entrada de Dados.
- **OBJETIVOS:**
  - O tipo `char`.
  - Função `rand()`
  - Dúvidas e Exercícios

Tipos primitivos: o tipo char

- O tipo `char` é utilizado para representar caracteres.
- Este tipo armazena um único caractere por vez.
- A máscara de leitura / escrita para variáveis char é `%c`.
- Quando um caractere é exibido na tela, ele é representado por um símbolo gráfico associado a seu valor numérico.
- Os símbolos gráficos e seus valores associados estão definidos na Tabela ASCII (American Standard Code for Information Interchange).

Tabela ASCII

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

Tipos primitivos: o tipo char

- Exemplo: lendo e imprimindo uma variável do tipo `char`.

```
1.  #include <stdio.h>
2.
3.  int main() // Funcao principal
4.  {
5.      // Variavel char: guarda um caractere
6.      char letra;
7.
8.      printf("Entre com um caractere:\n");
9.      scanf("%c", &letra);
10.
11.     printf("Voce digitou o caractere: %c\n", letra);
12.
13.     return 0;
14. } // Fim
```

Tipos primitivos: o tipo char

- Exemplo: lendo e imprimindo uma variável do tipo `char`.

```
1.  #include <stdio.h>
2.
3.  int main() // Funcao principal
4.  {
5.      // Variavel char: guarda um caractere
6.      char letra;
7.
8.      printf("Entre com um caractere:\n");
9.      scanf("%c", &letra);
10.
11.     printf("Voce digitou o caractere: %c\n", letra);
12.
13.     return 0;
14. } // Fim
```

Por enquanto não há muito mais o que se fazer com variáveis do tipo `char`.

Mais para frente, no decorrer do curso, podemos tomar decisões usando seus valores. Por exemplo:
Opção: S (sim) / N (não)
Sexo: M (masc) / F (fem)
Tamanho de roupa: P, M, G

Tipos primitivos: o tipo char

- Variáveis do tipo `char` são inicializadas dentro de aspas simples:

```
1.   #include <stdio.h>
2.
3.   int main()
4.   {
5.       char letra = ‘A’;
6.
7.       printf(“Caractere: %c\n”, letra);
8.
9.       return 0;
10.  }
```

Tipos primitivos: o tipo char

- Embora a Tabela ASCII contenha os dígitos de 0 a 9, não é possível realizar operações aritmética com esses dígitos quando armazenados em variáveis do tipo `char`.

```
1.   include...
2.   int main()
3.   {
4.        char c1 = ‘6’;
5.        char c2 = ‘5’;
6.        char c3 = c1 + c2; // Se c3 for impressa, a resposta nao eh 11!!!
7.        printf... // c3
8.
9.        return 0;
10.  }
```

- No exemplo acima, `c1` e `c2` guardam o símbolo associado aos valores 6 e 5, e não essas quantidades numéricas.

Geração de Números Aleatórios

- Números aleatórios são essenciais em praticamente todos os ramos da Computação.
  - Por exemplo: em jogos que você joga contra os inimigos, como eles sempre aparecem em posições e lugares diferentes?
  - Simples! Eles aparecem aleatoriamente.

- A função `rand()` é usada para gerar números aleatórios em C.

- Esta função pertence ao arquivo de cabeçalho `stdlib.h`.

- Os números gerados são inteiros (`int`) entre zero e 2147483647.

Geração de Números Aleatórios

- Exemplo: gerando dois números aleatórios e imprimindo seus valores.

Geração de Números Aleatórios

- Exemplo: gerando dois números aleatórios e imprimindo seus valores.

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.
4.  int main()
5.  {
6.      int n1, n2;
7.
8.      n1 = rand();
9.      n2 = rand();
10.
11.     printf("n1 = %d\n", n1);
12.     printf("n2 = %d\n", n2);
13.
14.     return 0;
15. }
```

Geração de Números Aleatórios

- Exemplo: gerando dois números aleatórios e imprimindo seus valores.

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.
4.  int main()
5.  {
6.      int n1, n2;
7.
8.      n1 = rand();
9.      n2 = rand();
10.
11.     printf("n1 = %d\n", n1);
12.     printf("n2 = %d\n", n2);
13.
14.     return 0;
15. }
```

Execute este código mais vezes e observe o resultado.

Geração de Números Aleatórios: seed

- A função `rand()` tem uma memória interna que armazena o inteiro, digamos `n`, produzido pela execução anterior da função.
- A cada nova execução, a função usa `n` para calcular um novo inteiro aleatório.
- O inteiro `n` que corresponde à primeira invocação de `rand()` é conhecido como semente (*seed*).
- O programador pode especificar a semente por meio da função `srand()` da biblioteca `stdlib.h`, que recebe a semente (um `int`) como argumento.
- Se o programador não especificar a semente, o sistema adota o valor zero como semente.

Geração de Números Aleatórios: seed

- Para deixar a sequência gerada imprevisível, basta iniciar a semente com um valor proveniente do relógio do sistema, através da função `time(NULL)`, pertencente ao arquivo de cabeçalho `time.h`.

Geração de Números Aleatórios: seed

- Para deixar a sequência gerada imprevisível, basta iniciar a semente com um valor proveniente do relógio do sistema, através da função `time(NULL)`, pertencente ao arquivo de cabeçalho `time.h`.

```
1.   #include <stdio.h> // printf
2.   #include <stdlib.h> // rand, srand
3.   #include <time.h> // time(NULL)
4.
5.   int main()
6.   {
7.       int n;
8.
9.       srand( time(NULL) );
10.
11.      n = rand();
12.      printf("n = %d\n", n);
13.
14.      return 0;
15.  }
```

Geração de Números Aleatórios: seed

- Para deixar a sequência gerada imprevisível, basta iniciar a semente com um valor proveniente do relógio do sistema, através da função `time(NULL)`, pertencente ao arquivo de cabeçalho `time.h`.

```
1.   #include <stdio.h> // printf
2.   #include <stdlib.h> // rand, srand
3.   #include <time.h> // time(NULL)
4.
5.   int main()
6.   {
7.       int n;
8.
9.       srand( time(NULL) );
10.
11.      n = rand();
12.      printf("n = %d\n", n);
13.
14.      return 0;
15.  }
```

Apenas uma vez no código, antes de usar `rand()`

Geração de Números Aleatórios

- E se quiséssemos trocar o intervalo de números gerado pela função `rand()`?
- Por exemplo:
- Como gerar valores no intervalo de 0 a 100?
- Como gerar valores no intervalo de 10 a 100?
- Como gerar valores no intervalo de –5 a 10?

Dúvidas?

Aula 4:
Caracteres e Números Aleatórios

***Disciplina*: Fundamentos de Programação**

Prof. Luiz Olmes

olmes@unifei.edu.br

UNIFEI
UNIVERSIDADE FEDERAL DE ITAJUBÁ