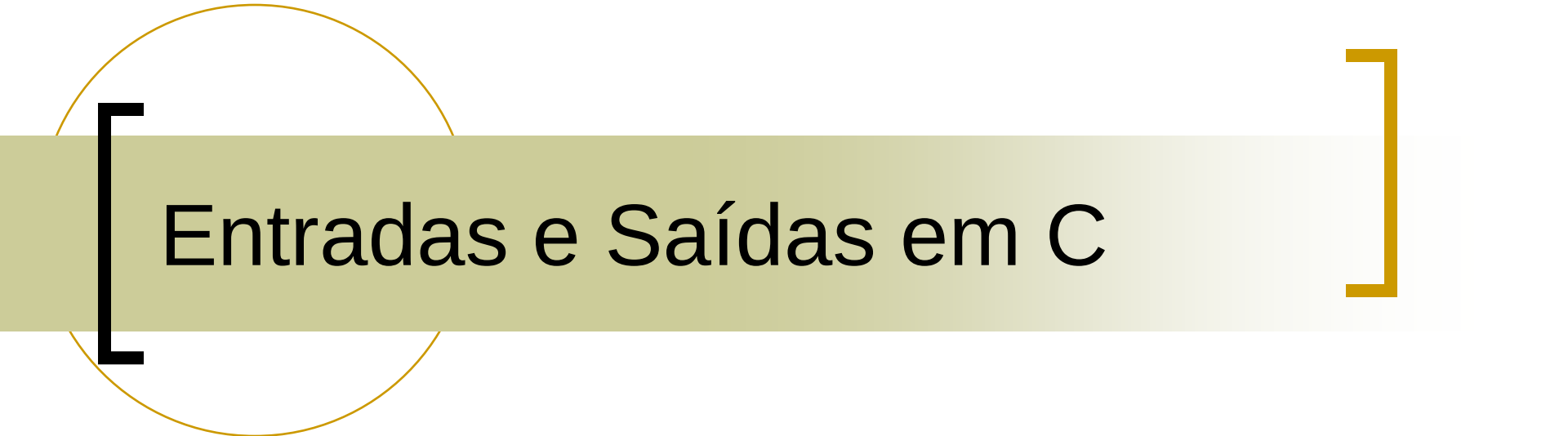




Entradas e Saídas em C

Luís Charneca

luis.charneca@gmail.com

[CARACTERES (1)]

- Os caracteres são um tipo de dados: o **char** e o C trata os caracteres como sendo variáveis de um byte (8 bits), ou seja ocupa um byte de memória.
- Um bit é a menor unidade de armazenamento de informações num computador.
- Exemplo de programa que usa caracteres:

```
#include <stdio.h>

int main ()
{
    char Ch;
    Ch='D';
    printf ("%c",Ch);
    return(0);
}
```

[CARACTERES (2)]

- No programa anterior, **%c** indica que `printf()` deve colocar um caracter no ecrã. Um `char` também é usado para armazenar um **número inteiro**. Este número é conhecido como o código **ASCII** correspondente ao caracter, como veremos no exemplo seguinte

[CARACTERES (3)]

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

```
int main ()
```

```
{
```

```
    char Ch;
```

```
    Ch='D';
```

```
    printf ("%d",Ch); /* Imprime o caracter como inteiro */
```

```
    return(0);
```

```
}
```

- Este programa vai imprimir o número 68 no ecrã, que é o código ASCII correspondente ao caracter 'D' (d maiúsculo).

FUNÇÕES DE LEITURA DE CARACTERES

Função	Forma Geral	Explicação
scanf	scanf ("%c", &variável);	Coloca o caracter inserido no identificador variável imprimindo-o no ecrã. O sistema só lê o caracter depois de pressionada a tecla enter .
getch	variável = getch();	Retorna um caracter para o identificador variável sem o imprimir no ecrã sem ser necessário premir a tecla enter (pode ser encontrada na biblioteca conio.h).
getche	variável = getche();	Retorna um caracter para o identificador variável imprimindo-o no ecrã sem ser necessário premir a tecla enter (pode ser encontrada na biblioteca conio.h).

[Função getch() / scanf]

```
#include <stdio.h>
#include <conio.h>
int main ()
{
    char Ch;
    Ch=getch();
    printf ("Você
           pressionou a tecla
           %c",Ch);
    return(0);
}
```

```
#include <stdio.h>
int main ()
{
    char Ch;
    scanf("%c", &Ch);
    printf ("Você
           pressionou a tecla
           %c",Ch);
    return(0);
}
```

[STRINGS (1)]

- No C uma string é um vector de caracteres terminado com um caracter nulo. O caracter nulo é um caracter com valor inteiro igual a zero (código ASCII igual a 0). O **terminador nulo** também pode ser escrito usando a convenção de barra invertida do C como sendo `'\0'`. Embora o assunto vectores seja discutido posteriormente, são apresentados aqui os fundamentos necessários para se poder utilizar as strings.

[STRINGS (2)]

- Para declarar uma string usa-se o seguinte formato geral:
- **char nome_da_string[tamanho];**
- Isto declara um vector de caracteres (uma string) com número de posições igual a tamanho.
- Tem que se reservar um caracter para ser o terminador nulo (`\0`), tem que se declarar o comprimento da string como sendo, no mínimo, um caracter maior que a maior string que pretendemos armazenar.

[STRINGS (3)]

- Se for necessário ler uma string fornecida pelo utilizador pode-se usar a função **gets()**. A função **gets()** coloca o terminador nulo (`\0`) na string, quando se prime a tecla "Enter".

```
#include <stdio.h>
int main ()
{
    char string[100];
    printf ("Digite uma string: ");
    gets (string);
    printf ("\n\nVocê digitou %s",string);
    return(0);
}
```

[STRINGS (4)]

- Como as strings são vectores de caracteres, para se aceder a um determinado caracter de uma string, basta "indexar", ou seja, usa-se um índice para aceder ao caracter desejado dentro da string.
- Exemplo:
`printf("%c", str[1] );`
- Por quê se está a aceder à segunda letra e não à primeira?
- Na linguagem C, o **índice começa em zero**. Assim, a primeira letra da string sempre estará na posição 0. A segunda letra sempre estará na posição 1 e assim sucessivamente.

[STRINGS (5)]

- Exemplo que imprimirá a segunda letra da string "João" apresentada acima. De seguida, ele mudará esta letra e apresentará a string no final.

```
#include <stdio.h>
int main()
{
    char str[10] = "João";
    printf("\n\nString: %s", str);
    printf("\nSegunda letra: %c", str[1]);
    str[1] = 'U';
    printf("\nAgora a segunda letra é: %c", str[1]);
    printf("\n\nString resultante: %s", str);
    return(0);
}
```

Alguns dos códigos %

<i>Código</i>	<i>Significado</i>
<code>%d, %i</code>	Inteiro
<code>%f</code>	Real
<code>%c</code>	Caracter
<code>%s</code>	String
<code>%%</code>	Coloca % no ecrã

Alguns exemplos de printf() e o que eles exibem:

<i>Código</i>	<i>Output</i>
<code>printf ("Teste %% %")</code>	"Teste % %"
<code>printf ("%f",40.345)</code>	"40.345"
<code>printf ("Um caracter %c e um inteiro %d",'D',120)</code>	"Um caracter D e um inteiro 120"
<code>printf ("%s e um exemplo","Este")</code>	"Este e um exemplo"
<code>printf ("%s%d%%","Juros de ",10)</code>	"Juros de 10%"

[Exercícios]

- Escreva um programa que leia um caracter digitado pelo utilizador, imprima o caracter digitado e o código ASCII correspondente a este caracter.
- Escreva um programa que leia duas strings e as coloque no ecrã. Imprima também a segunda letra de cada string.