



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PIAUÍ
Campus Teresina - Central

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E
TECNOLOGIA DO PIAUÍ

CAMPUS TERESINA-CENTRAL

DIRETORIA DE ENSINO

Estrutura de Dados

**Compilador
scanf() e printf()
a biblioteca iostream
namespace**

Professora: Elanne na O. dos Santos

elannecristina.santos@gmail.com
elannecristina.santos@ifpi.edu.br

Linguagem C

- Podemos definir a linguagem C como sendo uma linguagem de programação robusta e multiplataforma, projetada para aplicações modulares de rápido acesso.
- Podendo ser considerada como uma linguagem de médio nível, pois possui instruções que a tornam ora uma linguagem de alto nível e estruturada como o Pascal, se assim se fizer necessário, ora uma linguagem de baixo nível pois possui instruções tão próximas da máquina, que só o Assembler possui.
- De fato com a linguagem C podemos construir programas organizados e concisos (como o Pascal), ocupando pouco espaço de memória com alta velocidade de execução (como o Assembler). Infelizmente, dada toda a flexibilidade da linguagem, também poderemos escrever programas desorganizados e difíceis de serem compreendidos (como usualmente são os programas em BASIC).
- Devemos lembrar que a linguagem C foi desenvolvida a partir da necessidade de se escrever programas que utilizassem recursos próprios da linguagem de máquina de uma forma mais simples e portátil que o assembler.

CARACTERÍSTICAS DA LINGUAGEM C

- Portabilidade entre máquinas e sistemas operacionais.
- Dados compostos em forma estruturada.
- Programas Estruturados.
- Total interação com o Sistema Operacional.
- Código compacto e rápido, quando comparado ao código de outras linguagem de complexidade análoga.
- Para compilar e executar nossos programas utilizaremos o ambiente Bloodshed Dev-C++, disponível gratuitamente no *link*:

<http://www.bloodshed.net/devcpp.html>

A ESTRUTURA BÁSICA DE UM PROGRAMA EM C

- Um programa em C consistem em uma ou várias “funções”. Vamos começar pelo menor programa possível em C:

```
main( ) {  
}
```

- Os comentários iniciam com o símbolo */** e se estendem até aparecer o símbolo **/*.
- A diretiva *#include* inclui o conteúdo de um outro arquivo dentro do programa atual, ou seja, a linha que contém a diretiva é substituída pelo conteúdo do arquivo especificado.

Sintaxe:

```
#include <nome do arquivo>
```

A ESTRUTURA BÁSICA DE UM PROGRAMA EM C

- A tabela a seguir apresenta alguns dos principais .h da linguagem C:

Arquivo----Descrição

stdio.h ->Funções de entrada e saída (I/O)

string.h ->Funções de tratamento de strings

math.h ->Funções matemáticas

ctype.h ->Funções de teste e tratamento de caracteres

stdlib.h ->Funções de uso genérico

COMANDOS BÁSICOS - INSTRUÇÕES DE ENTRADA E SAÍDA

A FUNÇÃO PRINTF()

- É um dos mais poderosos recursos da linguagem C, printf() servirá basicamente para a apresentação de dados no monitor.
Sua forma geral será: printf(“string de controle”, lista de argumentos);

Ex.:

Exemplo: Dado um número, calcule seu quadrado.

```
#include<stdio.h>

main() {
    int numero;
    numero=10;
    printf(“O %d elevado ao quadrado resulta em %d. \n”,
        numero,numero*numero);
}
```

OBS: A diretiva #include foi utilizada, pois usamos o comando printf (stdio.h)

Usando o “gcc” para compilar no prompt de comando

- Com o gcc já instalado no seu sistema, é muito simples usá-lo para compilar programas em C. Se o programa consistir de um único arquivo, você pode simplesmente executar este comando no terminal:

```
gcc prog.c -o prog
```

- onde “*prog.c*” é o nome do arquivo que contém o código. Os outros dois parâmetros, “*-o prog*”, indicam o arquivo de saída do compilador e o arquivo executável que conterá o programa.

Usando o “gcc” ou “g++” para compilar no prompt de comando

Compilar:

```
gcc prog.cpp -o prog
```

Para incluir “iostream” do C++:

```
gcc prog.cpp -o prog -lstdc++
```

ou

```
g++ prog.cpp -o prog
```

- Executar : ***./prog***

Obs: gcc é o compilador C. g++ é o compilador do C++.

COMANDOS BÁSICOS - INSTRUÇÕES DE ENTRADA E SAÍDA

A FUNÇÃO SCANF()

- Uma das mais importantes e poderosas instruções, servirá basicamente para promover leitura de dados (tipados) via teclado.
- Sua forma geral será: `scanf("string de controle", lista de argumentos);`
- Algumas leituras básicas:
 - `%c` - leitura de caracter
 - `%d` - leitura de números inteiros
 - `%f` - leitura de números reais
 - `%s` - leitura de caracteres
- Cada variável a ser lida, deverá ser precedida pelo caracter `&`, por razões que no momento não convém explicarmos, mas que serão esclarecidas no decorrer do curso. Para seqüência de caracteres (`%s`), o caracter `&` não deverá ser usado.

COMANDOS BÁSICOS - INSTRUÇÕES DE ENTRADA E SAÍDA

A FUNÇÃO SCANF()

- Exemplo: Programa para ler e mostrar uma idade

```
/* Exemplo Lê e Mostra Idade */
```

```
main() {  
    int idade;  
    char nome[30];  
    printf("Digite sua Idade: ");  
    scanf("%d",&idade);  
    printf("Seu Nome: ");  
    scanf("%s",nome); /* Strings não utilizar '&' na leitura */  
    printf("%s sua idade e' %d anos. \n", nome, idade);  
}
```

A biblioteca *iostream*

- A **biblioteca** padrão de entrada e saída do C++ é a **iostream**.
- É preciso incluir o comando **using namespace** std. Este comando serve para definir um "espaço de nomes", ou **namespace**. Isso evita duplicidade com, por exemplo, outras implementações com nomes semelhantes. Ex.:

```
#include <iostream>
using namespace std;
int valor;
int main()
{ cout << "Exemplo de saída na tela" <<
  endl;
  cout<<"digite o valor :";
  cin>>valor; }
```

A biblioteca *iostream* e o tipo *string*

Exemplo

```
#include <iostream>
```

```
using namespace std;
```

```
int mat;
```

```
float nota;
```

```
char nome[30];
```

```
int main()
```

```
{ cout<<"Digite a matricula:";
```

```
cin>>mat;
```

```
cout<<"Digite a nome:";
```

```
cin>>nome;
```

```
cout<<"Digite a nota:";
```

```
cin>>nota;
```

```
cout<<mat<<"--"<<nome<<"--"<<nota;
```

```
}
```

Usando a *iostream* você pode usar também o tipo *string* para definir suas variáveis. Observe que neste exemplo a cadeia de caracteres está sendo lida com o char de 30 posições. No próximo exemplo vamos definir nome como uma string.

A biblioteca *iostream* e o tipo *string*

Exemplo

```
#include <iostream>
using namespace std;
int mat;
float nota;
string nome;
int main()
{
    cout<<"Digite a matricula:";
    cin>>mat;
    cout<<"Digite a nome:";
    cin>>nome;
    cout<<"Digite a nota:";
    cin>>nota;
    cout<<mat<<"--"<<nome<<"--"<<nota;
}
```

A biblioteca *iostream* e o tipo *string*

No estilo da linguagem C quando queremos representar um conjunto de caracteres colocamos todos eles em uma matriz sequenciada na memória:

Endereço relativo	0x0	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09
Dado	U	m	a		f	r	a	s	e	\0

Por exemplo, para declarar um espaço na memória que contenha 20 caracteres fazemos:

```
char dados[20];
```

A biblioteca *iostream* e o tipo *string*

Para manipular este tipo de string a biblioteca padrão da linguagem C dispõe de diversas funções (Ex.:*strcmp()*,*strcat()* etc), para mais detalhes consulte o livro [Programar em C](#).

No estilo C++, como era de se esperar, as strings são objetos, eles podem ser criados facilmente através da biblioteca padrão referenciada pelo arquivo de cabeçalho `<string>`. As strings são objetos com recursos que permitem manipular os seus caracteres com as funcionalidades das funções da linguagem C e mais algumas características próprias possibilitadas pela orientação a objetos.

Obs: link sugerido: https://pt.wikibooks.org/wiki/Programar_em_C

Namespaces

- Para usar os recursos de entrada e saída da biblioteca *iostream* em C++, é preciso incluir o comando **using namespace std**.
- Este comando serve para definir um "espaço de nomes", ou **namespace**. Um *namespace* permite a definição de estruturas, classes, funções, constantes, etc, que estarão vinculadas a ele. Por definição, a linguagem C++ utiliza o *namespace std* para definir todas as funções da biblioteca padrão.

Namespaces

- Se não utilizarmos o comando **using...**, será necessário especificar explicitamente o *namespace* utilizado, como por exemplo:

```
#include <iostream>

int main()
{
    std::cout << "Exemplo de saída na
tela" << std::endl;

    ...

}
```

Variáveis

Tabela de Tamanhos e Escala de Tipos Básicos

Tipo	Extensão	Escala Numérica em bits
Char	8	0 a 255
Int	16	-32768 a 32767
Float	32	3.4E-38 a 3.4E+38
Double	64	1.7E-308 a 1.7E+308
Void	0	sem valor

Máscaras

Máscara	Tipo de dado	Descrição
%d	Int	Mostra um número inteiro
%c	Char	Mostra um caractere
%f	Float ou double	Mostra um número decimal
%s	Char	Mostra uma cadeia de caracteres (string)
%i	Int	Mostra um inteiro
%ld	Long int	Mostra um número inteiro longo

Máscaras

As máscaras

Agora que vimos para que servem as máscaras, veremos qual é a máscara de cada tipo de entrada e saída.

máscara	tipo de dado	descrição
%d	int	mostra um número inteiro
%c	char	mostra um caracter
%f	float ou double	mostra um número decimal
%i	int	mostra um número inteiro
%ld	long int	mostra um número inteiro longo
%e	float ou double	mostra um número exponencial (número científico)
%E	float ou double	mostra um número exponencial (número científico)
%o	int	mostra um número inteiro em formato octal
%x	int	mostra um número inteiro em formato hexadecimal
%X	int	mostra um número inteiro em formato hexadecimal
%s	char	mostra uma cadeia de caracteres (string)

Variáveis

- Exemplo : Mesmo número com 2 representações diferentes.

```
main() {  
    float a;  
    printf("Digite um numero: ");  
    scanf("%f",&a);  
    printf("%f %e",a,a);  
}
```

Simulando obtemos:

```
    Digite um numero: 65  
65.000000 6.500000E+01
```

Variáveis

- Exemplo : Criando três variáveis e inicializando-as em tempo de criação.

```
Main ( ) {  
    int    evento  = 5;  
    char   corrida = 'c';  
    float  tempo   = 27.25;  
    printf (" o melhor tempo da eliminatória % c", corrida);  
    printf (" \n do evento %d  foi % f", evento, tempo);  
}
```

Simulando obtemos:

o melhor tempo da eliminatória c
do evento 5 foi 27.25

OPERADORES ARITMÉTICOS

Operador	Ação
+	Adição
*	Multiplicação
/	Divisão
%	Resto de Divisão Inteira
-	Subtração o menos unário
--	Decremento
++	Incremento

Exemplo

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    float nota1;
    float nota2;
    float nota3;
    float media;
    printf("\n Digite a primeira nota..: ");
    scanf("%f",&nota1);
    printf("\n Digite a segunda nota...: ");
    scanf("%f",&nota2);
    printf("\n Digite a terceira nota...: ");
    scanf("%f",&nota3);
    media=(nota1+nota2+nota3)/3;
    printf("\n\n Sua média .....:%6.2f",media);

}
```


OPERADORES RELACIONAIS E LÓGICOS

Operador	Ação
>	Maior que
>=	Maior ou igual que
<	Menor que
<=	Menor ou igual que
==	Igual a
!=	Diferente de
&&	Condição “E”
	Condição “OU”
!	Não