Algoritmos e Estruturas de Dados

Funções de entrada e saída

C

Funções de entrada e saída

As funções de entrada e saída realizam a conexão do algoritmo com o ambiente externo, recebendo e enviando dados.

Saída de dados

A saída de dados que utilizaremos aqui é a tela. Para mostrar uma mensagem na tela, utilizaremos a função abaixo, acompanhada da mensagem que desejamos apresentar na tela.

```
printf();
```

Atenção

Para que a função de saída `printf()` possa ser utilizada é necessário importar a biblioteca de entrada e saída padrão.

Para tal, basta incluir no início do código a linha

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

Exemplo

- Para apresentar na tela a mensagem `Olá Mundo`:

```
printf("Olá Mundo");
```

Saída na tela:

```
Olá Mundo
```

Atenção

Perceba que aqui estamos observando apenas a função de saída. Para que ela possa funcionar de maneira correta no algoritmo, é necessário que ela esteja dentro de uma **estrutura básica**.

```c
#include <stdio.h>
int main(void){
  printf("Olá Mundo\n");
  return 0;
}
```

Saída na tela:

```
Olá Mundo
```

Exemplo

- Para apresentar a mensagem `Bom dia, José`:

```
printf("Bom dia, José\n");
```

Saída na tela:

```
Bom dia, José
```

Exercício

- Desenvolva um algoritmo que apresenta seu nome na saída padrão.
- Para apresentar um valor específico:

```
printf("%d\n", 18);
```

Saída na tela:

```
18
```

- Pode-se também apresentar diversos resultados de uma única vez:

```
printf("%s %d %s\n", "A idade mínima deve ser de", 18, "anos");
```

Saída na tela:

```
A idade mínima deve ser de 18 anos
```

Uso de armazenamento intermediário

No desenvolvimento de algoritmos, podemos utilizar armazenamento intermediário de valores. Para tal, devemos indicar qual é o **tipo de dado** que deverá ser guardado e um **identificador** desse armazenamento.

Informação

O uso desse armazenamento é um tópico de grande importância no desenvolvimento de algoritmos. O veremos de forma mais detalhada ao abordarmos tipos primitivos de dados, constantes e variáveis.

Exemplo

- Armazenando um valor e apresentando na tela

```c
int altura; //identificador que permite armazenar um número inteiro
altura = 50;
printf("%d", altura); // saída: 50
```

Saída na tela:

```
50
```

É possível também apresentar diversos valores de uma única vez em uma única saída.

Exemplo

```
char nome = "Dunga";
int idade;

idade = 35;
printf("%s %s %d %s",nome,"tem", idade, "anos de idade.");
```

Saída na tela:

```
Dunga tem 35 anos de idade
```

Exercícios

1. Desenvolva um algoritmo que mostra na tela a mensagem *Hoje é dia de aula de Algoritmos*.

2. Desenvolva um algoritmo que dentro dele possui um armazenamento intermediário de caracteres chamado *mes_ferias_1* e *mes_ferias_2*, onde cada um deve possuir os valores *janeiro* e *julho*. Seu algoritmo deve informar na tela uma mensagem informando que estes são os meses de férias previsto no calendário acadêmico.

3. Desenvolva um algoritmo que dentro dele possui os armazenamentos intermediários de caracteres chamado *mes_ferias_1* e *mes_ferias_2*, onde cada um deve possuir os valores *janeiro* e *julho*. Também possui os armazenamentos intermediários *dias_ferias_1* e *dias_ferias_2*, com os conteúdos *30* e *15*, respectivamente. Seu algoritmo deve informar na tela uma mensagem informando que são previstos 30 dias de férias em janeiro e 15 dias de férias em julho.

Entrada de dados

Para que a entrada de dados possa ser realizado é necessário passar ao algoritmo uma informação adicional, com o identificador em que o valor deverá ser armazenado.

Para a entrada de dados pode-se utilizar a seguinte função:

```
gets(); // entrada de dados do tipo caractere
```

ou

```
scanf("%d", &); // entrada de dados do tipo inteiro
```

Atenção

Assim como na função de saída `printf`, para que as funções de entrada `scanf()` e `gets()` possam ser utilizadas também é necessário importar a biblioteca de entrada e saída padrão.

Conhecendo a função que realiza a leitura de dados da entrada padrão, devemos informar qual identificador será responsável por armazenar o dado recebido na entrada.

Exemplo

- recebendo valores na entrada e armazenando

recebendo da entrada um dado do tipo inteiro

```c
int numero;
scanf("%d", &numero); // observe o &
```

recebendo da entrada um dado do tipo caractere

```c
char palavra[100];
gets(palavra); //perceba que o & não é necessário aqui
```

Atenção

O uso da função `scanf()` requer cuidado ao mencionar o identificador de onde o valor será armazenado. Observe o uso do `&` antes do nome do identificado. Para caracteres com a função `gets()` este sinal não é necessário.

Estas diferenças e porque isto é realizado desta forma será explicado futuramente, quando os conteúdos de vetores e manipulação de cadeias de caracteres forem abordados.

Exemplo

- Solicite ao usuário que digite seu nome

```
char nome[100];
// highlight-next-line
gets(nome); //recebe o dado da entrada
```

<details> <summary>Código completo</summary>

```
#include <stdio.h>

int main(){
  char nome[100];
// highlight-next-line
  gets(nome); //recebe o dado da entrada

  return 0;
```

Exemplo

- Pergunte ao usuário seu nome e idade. Em seguida, mostre na tela as informações digitadas.

```c
//Variáveis
char nome[100];
int idade;

//Entrada
gets(nome);
scanf("%d", &idade)

//Saída
printf("%s %s %d %s\n", nome, "tem", idade, "anos de idade")
```

<details> <summary>Código completo</summary>

Exercícios

1. Solicite ao usuário que digite um número. Em seguida, mostre na tela o número digitado.
2. Desenvolva um algoritmo que pergunta ao usuário o nome do usuário, e em seguida responde "Boa noite, `user`", substituindo `user` pelo nome digitado.
3. Faça um algoritmo que pergunta ao usuário a sua idade, e em seguida informa a mensagem "Você tem `X` anos", substituindo `X` pela idade digitada.