Algoritmos e Programação B

Introdução

Algoritmos e Programação B

- **Objetivos**
  - desenvolver a lógica na resolução de problemas;
  - usar o raciocínio lógico para a programação de computadores.

Usar o raciocínio lógico requer

- calma,
- conhecimento,
- experiência...

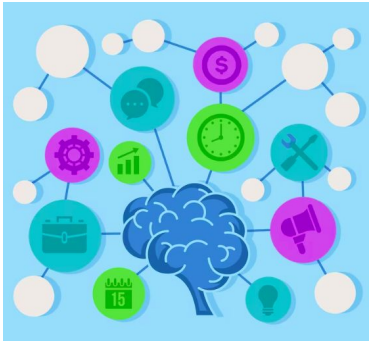
Usar o raciocínio lógico requer

- criatividade,
- responsabilidade,
- persistência...

Algoritmos e Programação B

Aperfeiçoar o raciocínio,

para resolução de problemas!!



Algoritmos e Programação B

Conjuntos Heterogêneos

Conjuntos Heterogêneos

- Conjuntos Heterogêneos constituem tipos de dados definidos pelo usuário:
  - Estruturas (*struct*)
  - Uniões (*union*)
  - Enumeração (*enum*)

- e ainda o uso de
  - *typedef*

Estrutura

- Uma estrutura é uma coleção de variáveis agrupadas sob um mesmo nome, sendo uma forma de manter e utilizar dados inter-relacionados.
- Os membros de uma estrutura são chamados elementos ou campos.
- Palavra-chave `struct` informa ao compilador que um modelo de estrutura está sendo definido.

Estrutura

- Declaração do **tipo de dado**

```
struct especificador_de_tipo{
        Tipo membro1;
        Tipo membro1;
        Tipo membro1;
        ...
};
```

```
struct TipoAluno{
        char matricula[11];
        char nome[30];
        int anoIngresso;
        float mediaVestibular;
        ...
};
```

Estrutura

- Declaração de **variável** do tipo de dado criado:

```
struct especificador_de_tipo nome_da_variável;
```

```
struct TipoAluno a;
```

- Quando a variável é declarada, o compilador alocará memória para a estrutura conforme ela foi definida na declaração do tipo de dado com `struct`.

Tamanho dos tipos de dados

Tipo de dado	Tamanho
char	1 byte
int [-32.767 a 32.767]	2 bytes
unsigned int [0 a 65.535]	2 bytes
long int	4 bytes
float (6 dígitos)	4 bytes
double (10 dígitos)	8 bytes
long double (10 dígitos)	10 bytes

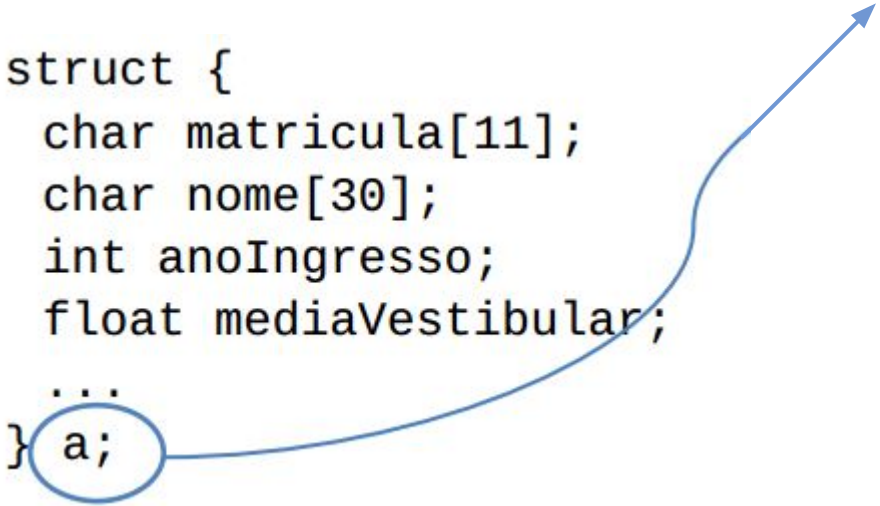
Estrutura

- Podem ser criadas uma ou mais **variáveis** com o tipo de dado especificado por `struct`.

```
struct TipoAluno a, b;
```

Estrutura

- Se apenas uma variável é necessária, o nome da estrutura não é necessário. Então a declaração pode ser feita apenas com o nome da variável:



Referenciando elementos da estrutura

- Elementos individuais (campos) são referenciados por meio do operador ponto (.).
- Para ler elementos de uma estrutura, como a do exemplo:

```
gets(a.nome);

gets(a.matrícula);

scanf("%d",&a.anoIngresso);
```

Como mostrar os valores dos elementos da estrutura?

- Para mostrar na tela elementos de uma estrutura, como a do exemplo:

```
puts(a.nome);

puts(a.matrícula);

printf("Ano de ingresso: %d\n",a.anoIngresso);
```

Exemplo: Definição da struct e declaração de variável

```c
#include <stdio.h>

struct TipoAluno{
    char matricula[11];
    char nome[30];
    int anoIngresso;
    float mediaVestibular;
};

int main() {
    struct TipoAluno aluno;

```

Exemplo: leitura dos elementos de uma estrutura

```
13      printf("Digite a matrícula (máximo 10 caracteres): ");
14      scanf("%s", aluno.matricula);
15
16      printf("Digite o nome do aluno: ");
17      scanf(" %s", aluno.nome);
18
19      printf("Digite o ano de ingresso: ");
20      scanf("%d", &aluno.anoIngresso);
21
22      printf("Digite a média do vestibular: ");
23      scanf("%f", &aluno.mediaVestibular);
```

Exemplo: exibição de elementos de uma estrutura

```
24
25      printf("\nDados do aluno:\n");
26      printf("Matrícula: %s\n", aluno.matricula);
27      printf("Nome: %s\n", aluno.nome);
28      printf("Ano de Ingresso: %d\n", aluno.anoIngresso);
29      printf("Média do Vestibular: %.2f\n", aluno.mediaVestibular);
30
31      return 0;
32  }
```

Como mostrar uma string, caractere a caractere na tela?

- Para mostrar na tela, por exemplo, o nome do aluno, caracter a caracter, podemos usar as funções `putchar()` ou `printf()` em um laço de repetição:

```
for(i = 0; a.nome[i] != "\0"; i++){
   printf("%c\n", a.nome[i]);
}
```

Matrizes ou Vetores de Estruturas

Matrizes ou vetores de estrutura

- Podemos criar uma matriz ou vetor de estruturas.
- Primeiramente, é necessário definir a estrutura e depois declarar o vetor ou matriz do tipo definido.
- Por exemplo: para armazenar os valores de 100 alunos, é definido um vetor do tipo TipoAaluno:

```
struct TipoAluno a[100];
```

Tipo do dado

Vetor

Exemplo: struct e variável para 5 alunos

```
3   struct TipoAluno{
4       char matricula[11];
5       char nome[30];
6       int anoIngresso;
7       float mediaVestibular;
8   };
9
```

```
10  int main() {
11      struct TipoAluno alunos[5];
12      int i;
```

Exemplo: Leitura dos dados

```
14      for (i = 0; i < 5; i++) {
15          printf("Digite os dados do aluno %d:\n", i + 1);
16
17          printf("Matrícula (máximo 10 caracteres): ");
18          scanf("%s", alunos[i].matricula);
19
20          printf("Nome do aluno: ");
21          scanf(" %s", alunos[i].nome);
22
23          printf("Ano de ingresso: ");
24          scanf("%d", &alunos[i].anoIngresso);
25
26          printf("Média do vestibular: ");
27          scanf("%f", &alunos[i].mediaVestibular);
28
29          printf("\n");
30      }
```

Exemplo: Exibição dos dados

```
33      printf("\nDados dos alunos:\n");
34      for (i = 0; i < 5; i++) {
35          printf("Aluno %d:\n", i + 1);
36          printf("Matrícula: %s\n", alunos[i].matricula);
37          printf("Nome: %s\n", alunos[i].nome);
38          printf("Ano de Ingresso: %d\n", alunos[i].anoIngresso);
39          printf("Média do Vestibular: %.2f\n\n", alunos[i].mediaVestibular);
40      }
41
42
43      return 0;
44  }
```