Laboratório de Algoritmos I
Novos Tipos de Dados em C: struct e typedef

Cast

- Antes de começarmos o assunto desta aula, é interessante aprendermos o conceito de “cast”.

Cast

- “Cast” significa converter um tipo primitivo em outro. Por exemplo, converter um valor double para int.
- Para isso, a sintaxe é simples: colocar o tipo desejado entre parênteses antes do valor a ser convertido. Exemplo:

```
double teste = 2.5;
int teste2 = (int) teste;
```

- A variável teste2 receberá o valor 2. (parte inteira)

Mais Funções da Biblioteca Padrão

- atoi(): recebe um string como parâmetro e o converte para um valor inteiro. Caso não contenha um número válido, retornará 0. Espaços em branco iniciais são ignorados.
- atof(): recebe um string como parâmetro e o converte para um valor double. Caso não contenha um número válido, retornará 0. Espaços em branco iniciais são ignorados.
- exit(): termina imediatamente o programa.

Introdução

- Problema: agrupar dados relacionados (mas de tipos diferentes) sob um mesmo nome.
- Solução em C: estruturas!
- Estruturas são tipos de variáveis que agrupam dados geralmente desiguais; ao passo que matrizes agrupam dados similares. Os itens de dados da estrutura são chamados de **membros**, e os da matriz, de **elementos**.

struct

- Por meio desta palavra-chave é que criamos uma estrutura em C, ou seja, definimos um novo tipo de dado.
- Definir um tipo de dado significa informar ao compilador seu nome, tamanho em bytes e forma como deve ser armazenado e recuperado da memória.

struct - Exemplo

```
struct Aluno
{
  int nmat;
  float nota[3];
  float media;
};
```

struct

- Esta definição pode ser escrita fora de qualquer função (acesso global), como dentro de uma função (acesso local).
- Após ter sido definido, o novo tipo existe e pode ser utilizado para criar variáveis de modo similar a qualquer tipo simples.
- Para acessar os membros de uma estrutura, deve ser utilizado o operador ponto (“.”).

struct – Exemplo de Uso

```
int main(){
  struct Aluno Jose;
  Jose.nmat = 456;
  Jose.nota[0] = 7.5;
  Jose.nota[1] = 5.2;
  Jose.nota[2] = 8.4;
  Jose.media = (Jose.nota[0] + Jose.nota[1] +
  Jose.nota[2])/3;
  printf("Matricula: %d\n", Jose.nmat);
  printf("Media: %.2f\n", Jose.media);
  return 0;
}
```

Novos nomes para tipos - typedef

- Declarações com **typedef** não criam novos tipos. Apenas criam sinônimos para tipos existentes.

Novos nomes para tipos - typedef

- Sintaxe:

```
typedef tipo-existente sinônimo;
```

- Exemplos:

```
typedef unsigned char BYTE;
typedef unsigned int uint;
```

typedef – declarando variáveis

- Exemplos:

```
int main(){
  BYTE ch;
  uint x;
  unsigned char chmm; /* os tipos originais
  continuam disponíveis*/
```

Usando typedef com struct

```
struct Aluno
{
  int nmat;
  float nota[3];
  float media;
};
typedef struct Aluno Aluno;
Aluno Jose; //declarando variável
```

Usando typedef com struct
Segunda possibilidade

```
typedef struct Aluno
{
  int nmat;
  float nota[3];
  float media;
} Aluno;

Aluno Jose; //declarando variável
```

Usando typedef com struct
Terceira possibilidade

```
typedef struct
{
  int nmat;
  float nota[3];
  float media;
} Aluno;

Aluno Jose; //declarando variável
```

Inicializando estruturas - exemplo

```
typedef struct
{
  int dia;
  char mes[10];
  int ano;
} Data;
Data natal = {25, “dezembro”, 2009};
```

Atribuições entre estruturas

- Uma variável estrutura pode ser atribuída a outra do mesmo tipo por meio de uma atribuição simples:

```
Data aniversario = {30, "julho", 2009};
Data Andre;
Andre = aniversario;
```

Operações entre estruturas

- Em linguagem C, operações simples como a soma não estão definidas para tipos criados com a palavra struct. A soma deve ser efetuada membro a membro.

Estruturas aninhadas

```
typedef struct
{
  int dia;
  char mes[10];
  int ano;
} Data;
typedef struct
{
  int pecas;
  float preco;
  Data diavenda;
} Venda;
```

Estruturas aninhadas

```
int main() {
  Venda A = {20, 110.0, {7, “novembro”, 2008} };
  printf(“Data: %d de %s de %d\n”,
  A.diavenda.dia, A.diavenda.mes,
  A.diavenda.ano);
  return 0;
}
```

Saída:

```
Data: 7 de novembro de 2008
```

Passando estruturas para funções

- Em linguagem C, as estruturas podem ser passadas como parâmetros de funções da mesma forma que variáveis dos tipos primitivos.

Matrizes de estruturas

- Uma matriz de estruturas pode ser declarada e utilizada da mesma forma que uma matriz de tipos primitivos.

```
Venda vendas[50];  /* declara uma matriz
  cujo rótulo é "vendas" com 50 posições
  para armazenar variáveis do tipo "Venda",
  definido anteriormente */
```