Notas de aula - 25/03

TADs - Tipos Abstratos de Dados

Tipo de dados: definem o domínio de uma variável: int, float, char…

Estrutura de dados: representa um relacionamento lógico entre tipos de dados. Ex.:

```
struct endereco {
      char rua[30];
      int numero;
      char bairro[30];
      char cep[9];
};
```

Temos variáveis do tipo char e int, mas que juntas têm um significado: representar o endereço de uma pessoa.

Outro exemplo:

```
struct produto {
      char nome[30];
      char marca[30];
      int codigo;
      float valor;
      int categoria; // produto de limpeza, produto de beleza, doces, salgados, etc.
};
```

Temos variáveis do tipo char, int e float, que juntas têm o propósito de representar um produto de um supermercado.

TADs - Tipos Abstratos de Dados: Em um TAD, além de ter uma estrutura de dados, tenho também um conjunto de operações para manipular esses dados.

Considere por exemplo que você tem uma struct que representa um aluno da disciplina de Fundamentos de programação:

```
struct aluno_fundamentos {
      char nome[30];
      int RA;
      float notas[3];
      float media;
};
```

Imagine que você precisa:

- Cadastrar 44 alunos que se matricularam na turma;
- Excluir 4 alunos que conseguiram equivalência da matéria;
- Cadastrar 2 alunos que entraram na segunda chamada;

- Guardar as notas das 3 provas de cada aluno;
- Calcular a média de cada aluno;

Todas as operações acima podem ser consideradas operações que manipulam a estrutura de dados aluno_fundamentos. A estrutura de dados aluno_fundamentos e o conjunto das suas operações foram um TAD.

TAD = estrutura de dados + operações de manipulação da estrutura.

Operações comuns em TADs

- Criação da estrutura;
- Inclusão de um elemento;
- Remoção de um elemento;
- Busca de um elemento;
- Atualização de um elemento;
- Imprimir dados;
- Etc.

Toda a manipulação da estrutura é feita por meio de funções que interagem com ela. Para cadastrar um novo aluno, por exemplo, você **não** deve fazer o seguinte:

```
main() {
      struct aluno_fundamentos aluno;
      char nome[30];
      int registro_academico;
      float notas[3];

      printf("Informe o nome do aluno: ");
      scanf("%[^\n]s", &nome);
      printf("Informe o RA do aluno: ");
      scanf("%d", &registro_academico);
      printf("Informe 3 notas do aluno: ");
      scanf("%f %f %f", &notas[0], &notas[1], &notas[2]);

      strcpy(aluno.nome, nome);
      aluno.RA = registro_academico;
      aluno.notas[0] = notas[0];
      aluno.notas[1] = notas[1];
      aluno.notas[2] = notas[2];
}
```

Você pode criar uma função chamada *cadastra_aluno*, por exemplo, que recebe as informações do aluno e então cadastra essas informações. Assim, sempre que você precisar cadastrar um aluno, você poderá usar esta função, em vez de acessar a estrutura de forma direta:

```
struct aluno_fundamentos cadastra_aluno(char nome[], int
registro_academico, float notas[]) {
	struct aluno_fundamentos novo_aluno;

	strcpy(novo_aluno.nome, nome);
	novo_aluno.RA = registro_academico;
	novo_aluno.notas[0] = notas[0];
	novo_aluno.notas[1] = notas[1];
	novo_aluno.notas[2] = notas[2];

	return novo_aluno;
}


main() {
	struct aluno_fundamentos aluno;
	char nome[30];
	int registro_academico;
	float notas[3];

	printf("Informe o nome do aluno: ");
	scanf("%[^\n]s", &nome);
	printf("Informe o RA do aluno: ");
	scanf("%d", &registro_academico);
	printf("Informe 3 notas do aluno: ");
	scanf("%f %f %f", &notas[0], &notas[1], &notas[2]);

	aluno = cadastra_aluno(nome, registro_academico, notas);
}
```

Para calcular a média de um aluno:
Errado, seguindo a abordagem de TAD:

```
main() {
	struct aluno_fundamentos aluno;
	char nome[30];
	int registro_academico;
	float notas[3];

	printf("Informe o nome do aluno: ");
	scanf("%[^\n]s", &nome);
	printf("Informe o RA do aluno: ");
	scanf("%d", &registro_academico);
```

```
	printf("Informe 3 notas do aluno: ");
	scanf("%f %f %f", &notas[0], &notas[1], &notas[2]);

	aluno = cadastra_aluno(nome, registro_academico, notas);

	// média atualizada acessando a estrutura diretamente
	aluno.media = (aluno.notas[0] + aluno.notas[1] + aluno.notas[2]) /
3;
}
```

Correto:

```
struct aluno_fundamentos calula_media(struct aluno_fundamentos aluno) {
	int i;
	float soma = 0;

	for(i = 0; i < 3; i++) {
	    soma += aluno.notas[i];
	}
	aluno.media = soma / 3;

	return aluno;
}

main() {
	struct aluno_fundamentos aluno;
	char nome[30];
	int registro_academico;
	float notas[3];

	printf("Informe o nome do aluno: ");
	scanf("%[^\n]s", &nome);
	printf("Informe o RA do aluno: ");
	scanf("%d", &registro_academico);
	printf("Informe 3 notas do aluno: ");
	scanf("%f %f %f", &notas[0], &notas[1], &notas[2]);

	aluno = cadastra_aluno(nome, registro_academico, notas);

	// atualização da média não é feita diretamente sobre a estrutura,
é feita por função
	aluno = calula_media(aluno);
}
```

Resumindo: o programa só deve acessar a estrutura por meio de funções, nunca diretamente.

Implementar estruturas pensando em TADs implica em:

Reutilização de código: sempre que for preciso fazer uma operação sobre a estrutura basta chamar a função responsável. Ex.: cada vez que eu precisar cadastrar um novo aluno, basta fazer uma chamada da função *cadastra_aluno*.

Em vez de escrever a lógica para cadastrar um aluno várias vezes (uma para cada aluno), eu preciso escrever a lógica apenas uma vez dentro de uma função e depois basta chamar essa função, escrevendo apenas uma linha de código para cada cadastro necessário.

Encapsulamento: quem usa o TAD precisa apenas conhecer suas funcionalidades e não como a estrutura foi implementada. Por exemplo: quando você importa a biblioteca string.h, você pode usar diversas funções: strlen (pegar o tamanho da string), strcpy (copiar o conteúdo de uma string em outra), strcmp (comparar duas strings), etc. Você consegue usar todas essas funcionalidades sem saber como a linguagem implementa uma string internamente.

Da mesma forma, se você criar seu próprio TAD e disponibilizar para outro programador utilizar, ele só vai precisar conhecer as funções que você disponibilizou e não como você criou a estrutura. A vantagem disso é que o usuário só conseguirá manipular a estrutura por meio das funções criadas por você, evitando alterações inesperadas na estrutura.

Exemplo de TAD Ponto

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

static struct ponto {
	float x;
	float y;
};
typedef struct ponto Ponto;


Ponto cria_ponto(float x, float y) {
	Ponto p;
	p.x = x;
	p.y = y;
	return p;
}

void imprime_ponto(Ponto p) {
	printf("%.2f, %.2f\n", p.x, p.y);
}
```

```c
void altera_ponto(Ponto *p, float x, float y) {
      p->x = x; //(*p).x = x;
      p->y = y; //(*p).y = y;
}

float calcula_distancia(Ponto p1, Ponto p2) {
      float dx, dy;
      dx = p2.x - p1.x;
      dy = p2.y - p1.y;
      return sqrt(pow(dx, 2) + pow(dy, 2));
}


main() {
   Ponto p1, p2;
   float distancia;

   p1 = cria_ponto(10, 15);
   imprime_ponto(p1);

   p2 = cria_ponto(10, 25);
   imprime_ponto(p2);

   altera_ponto(&p2, 10, 35);
   imprime_ponto(p2);

   distancia = calcula_distancia(p1, p2);
   printf("%.2f", distancia);
}
```

Se a ideia é deixar algumas coisas “escondidas”, porque tudo está no mesmo arquivo? Mais a frente veremos que por convenção, ao criar um TAD usamos dois arquivos: um arquivo .h só com protótipos de funções, tipos de ponteiros e variáveis globais, e um arquivo .c, onde ficam os tipos de dados e implementação de funções. O usuário terá acesso ao arquivo .h. Ele será capaz de importar esse arquivo assim como fazemos com a string.h, math.h ou outras bibliotecas.

Exercício

Criar um TAD Livro. Sua estrutura de dados deve guardar as informações:

- Título;
- Gênero;
- Ano;
- Valor.

Deve ser possível fazer as seguintes operações:

- Cadastrar livros;
- Ajustar o valor de todos os livros de acordo com um percentual;
- Imprimir os dados de um livro;
- Exibir os dados do livro mais barato;
- Exibir os dados do livro mais caro;
- Exibir os dados de todos os livros de um determinado gênero;
- Exibir os dados de todos os livros que são lançamento (livros com ano igual a 2019).

Links interessantes

Vídeo aulas prof. André Backes sobre TADs: https://www.youtube.com/watch?v=bryesHll0vY