Linguagem de Programação C

PROFESSOR: DIEGO RICARDO KROHL
diego.krohl@ifc.edu.br

Tipos Abstratos de Dados (TADs)

- Agrupa a estrutura de dados juntamente com as operações que podem ser feitas sobre esses dados;
- O TAD encapsula a estrutura de dados. Os usuários do TAD só tem acesso a algumas operações disponibilizadas sobre esses dados;
- Usuário do TAD x Programador do TAD;
  - Usuário só “enxerga” a interface, não a implementação.

Implementação de TADs

- Em linguagens orientadas a objeto (C++, Java) a implementação é feita através de classes;
- Em linguagens estruturadas (C, pascal) a implementação é feita pela definição de tipos juntamente com a implementação de funções;
- Conceitos de C (**structs** e **typedef**);

Revisando – Estruturas / registros (Structs)

- Um registro é uma coleção de uma ou mais variáveis colocadas juntas sob um único nome para manipulação conveniente;
- Por exemplo, para representar um aluno são necessárias as informações nome, matrícula, conceito;
- Ao invés de criar três variáveis, é possível criar uma única variável contendo três campos;
- Em C, usa-se a construção struct para representar esse tipo de dado.

Revisando – Estruturas / registros (Structs)

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

struct Aluno {
    char nome[100];
    int matricula;
    char conceito;
};

main() {
    struct Aluno al, aux;
    strcpy(al.nome, "Pedro");
    al.matricula = 200712;
    al.conceito = 'A';
    aux = al;
    printf("Nome: %s\n", aux.nome);
    printf("Matricula: %i\n", aux.matricula);
    printf("Conceito: %c\n", aux.conceito);
}
```

Declaração de tipos (typedef)

- Para simplificar, uma estrutura ou mesmo outros tipos de dados podem ser definidos como um novo tipo;
- Uso da construção **typedef:**

```
typedef struct {
    char nome[30];
    int matricula;
    char conceito;
} TipoAluno;

int main() {
 TipoAluno al;

 ...
}
```

Tipos Abstratos de Dados (TADs)

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

typedef struct {
    char nome[100];
    int matricula;
    char conceito;
} Aluno;

main() {
    Aluno al, aux;
    strcpy(al.nome, "Pedro");
    al.matricula = 200712;
    al.conceito = 'A';
    aux = al;
    printf("Nome: %s\n", aux.nome);
    printf("Matricula: %i\n", aux.matricula);
    printf("Conceito: %c\n", aux.conceito);
}
```

TADs em C

- Para implementar um Tipo Abstrato de Dados em C, usa-se a definição de tipos juntamente com a implementação de funções que agem sobre aquele tipo;
- Como boa regra de programação, evita-se acessar o dado diretamente, fazendo o acesso somente através das funções;
  - Mas, diferentemente de C++ e Java, não há uma forma de proibir o acesso;
  - Pode-se implementar TADS em arquivos separados do programa principal.

Implementação - Exemplo

Escreva um trecho de código para fazer a criação dos novos tipos de dados conforme solicitado abaixo:

- Horário: composto de hora, minutos e segundos;

- Data: composto de dia, mês e ano;

- Compromisso: composto de uma data, horário e texto que descreve o compromisso;

- Crie uma main para permitir criar compromissos;

Implementação - Exemplo

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Definição dos novos tipos de dados
typedef struct {
    int hora;
    int minutos;
    int segundos;
} Horario;

typedef struct {
    int dia;
    int mes;
    int ano;
} Data;

typedef struct {
    Data data;
    Horario horario;
    char descricao[100];
} Compromisso;
```

Implementação - Exemplo

```
24  // Função para ler um horário
25  Horario lerHorario() {
26      Horario h;
27      printf("Digite a hora (0-23): ");
28      scanf("%d", &h.hora);
29      printf("Digite os minutos (0-59): ");
30      scanf("%d", &h.minutos);
31      printf("Digite os segundos (0-59): ");
32      scanf("%d", &h.segundos);
33      return h;
34  }
35
36  // Função para ler uma data
37  Data lerData() {
38      Data d;
39      printf("Digite o dia: ");
40      scanf("%d", &d.dia);
41      printf("Digite o mês: ");
42      scanf("%d", &d.mes);
43      printf("Digite o ano: ");
44      scanf("%d", &d.ano);
45      return d;
46  }
```

Implementação - Exemplo

```
48  // Função para ler um compromisso
49  Compromisso lerCompromisso() {
50      Compromisso c;
51      c.data = lerData();
52      c.horario = lerHorario();
53      printf("Digite a descrição do compromisso: ");
54      scanf(" %99[^\n]", c.descricao);
55      // Lê até 99 caracteres ou até encontrar uma nova linha
56      // pode-se usar o fgets
57      //fgets(c.descricao, sizeof(c.descricao), stdin);
58      return c;
59  }
60
61  // Função para exibir um compromisso
62  void exibirCompromisso(Compromisso c) {
63      printf("Compromisso em %02d/%02d/%04d às %02d:%02d:%02d\n",
64             c.data.dia, c.data.mes, c.data.ano,
65             c.horario.hora, c.horario.minutos, c.horario.segundos);
66      printf("Descrição: %s\n", c.descricao);
67  }
```

Implementação - Exemplo

```
69  main() {
70      int n, i;
71      printf("Quantos compromissos deseja criar? ");
72      scanf("%d", &n);
73      Compromisso compromissos[n];
74
75      for (i = 0; i < n; i++) {
76          printf("\nCompromisso %d:\n", i + 1);
77          compromissos[i] = lerCompromisso();
78      }
79
80      printf("\nCompromissos criados:\n");
81      for (i = 0; i < n; i++) {
82          printf("\nCompromisso %d:\n", i + 1);
83          exibirCompromisso(compromissos[i]);
84      }
85  }
```