Laboratório de Programação II

Tipos Abstratos de Dados

Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação

Aula de Hoje

- Criação de TADs
- Classes
  - Definição
  - Implementação
  - Uso
- Aplicações com TADs

Revisão

- Para criar Tipos Abstratos de Dados (TADs) em C++ será utilizado o mecanismo de classes.
- Definição de uma classe (arquivo **.h**)

```
class NomeDoTAD
{
 private:
   // definicao dos atributos
   int atributo1;
   float atributo2;

 public:
   NomeDoTAD(int x, int y); // construtor
   ~NomeDoTAD();            // destrutor

   // definicao das operacoes
   float calculaValor(int n, int l);
};
```

Revisão

- Implementação da classe (arquivo **.cpp**)

```
NomeDoTAD::NomeDoTAD(int x, int y)
{
  // inicializa o objeto
  atributo1 = x*x;
  atributo2 = y;
}

NomeDoTAD::~NomeDoTAD() // destrutor
{
  // finaliza
  // libera memoria (se necessario)
}

float NomeDoTAD::calculaValor(int n, int l)
{
  float conta = n + sqrt(l);
  return conta;
}
```

Atenção

Para alguns exercícios existem projetos do CodeBlocks com algum código pronto, disponível no site do curso. Faça o download e utilize.

sites.google.com/site/edlab2ufjf/arquivoslab2

Exercícios

1. Considere o TAD Aluno definido pela seguinte classe:

```
class Aluno
{
 private:
   int idade;
   string nome, matricula;
   double notas[7];

 public:
   Aluno(string n, string mat);
   ~Aluno();

   // operacoes
   void leNotas();          // exercicio 1
   double calculaMedia(); // exercicio 1
};
```

Exercícios

Ponteiros

1. a) Implemente a função `leNotas()` que deve ler 7 valores reais, os quais representam as notas de um aluno em disciplinas diferentes.

   b) Implemente a função `calculaMedia()` que deve retornar qual a média do aluno em todas as disciplinas cursadas pelo mesmo.

   Obs: o programa principal (`main.cpp`) está funcionando corretamente, mesmo sem essas implementações. Após a sua implementação o programa irá exibir a média calculada corretamente.

Exercícios

2. No TAD `Aluno` já existem funções para alterar e retornar o nome do aluno. Entretanto, as funções para alterar e retornar os seguintes atributos ainda não foram implementadas:
   - idade
   - matricula

   Defina e implemente as funções necessárias para realizar essas operações.

```
class Aluno
{
 // ...
 public:
   // ...
   string getNome();
   void setNome(string n);

   // exercicio 2
   // get e set para idade e matricula
};
```

Exercícios

3. Utilize as funções implementadas no Exercício 2 para imprimir todos os dados de um aluno no programa principal (`main.cpp`).
4. Considere que agora seja preciso armazenar se o aluno foi frequente ou não em cada disciplina. Para isso, inclua na classe um vetor de valores booleanos (do mesmo tamanho do vetor de notas).
5. Implemente uma função para ler os dados da frequência do aluno em cada uma das disciplinas. Para valores `true` digite 1, para valores `false` digite 0.

Exercícios

6. Por fim, faça uma função que irá imprimir um relatório detalhado sobre o aluno, incluindo as seguintes informações:
   - Nome
   - Idade
   - Matricula
   - Para cada disciplina, imprima:
     - se o aluno foi frequente;
     - se ele foi aprovado ou não.

   Considere que para ser aprovado em uma disciplina é preciso ter nota ≥ 60.

Exercícios

7. TAD `Prova`. Defina e implemente um TAD para representar uma prova, a qual deve conter os seguintes dados:
   - número de questões N;
   - um vetor com N valores reais para representar a nota de cada questão (esse vetor deve ser alocado de forma dinâmica no construtor);
   - um número real para armazenar a nota final.

   Além disso, o TAD deve oferecer as seguintes operações:
   - construtor: recebe N como parâmetro e alocar o vetor;
   - destrutor: libera memória do vetor;
   - ler as notas de cada questão;
   - calcular a nota final;
   - obter a nota final;

   A nota final será calculada somando-se a nota de cada questão, sem considerar a menor nota obtida em alguma questão.

Exercícios

Ponteiros

7. TAD `Prova`.

```
class Prova
{
 private:
   int n;
   double notaFinal;

   // ponteiro para vetor das notas
   // esse vetor deve ser alocado dinamicamente
   double *notasQuestoes;

public:
  Prova(int nq);
  ~Prova();
  void leNotas();
  void calculaNotaFinal();
  double obtemNotaFinal();
};
```