Algoritmos e Estrutura de Dados II

Tipos Abstratos de Dados

prof. Frederico Santos de Oliveira

Universidade Federal de Mato Grosso
Faculdade de Engenharia

Agenda

Qual a diferença entre um algoritmo e um programa?

Introdução

- Algoritmo
  - Sequência de ações executáveis para a solução de um determinado tipo de problema "apontando" para a variável que contém o dado desejado.
  - Exemplo: uma receita de bolo.
  - Em geral, algoritmos trabalham sobre estruturas de dados.
- Estruturas de Dados
  - Conjunto de dados que representa uma situação real.
  - Abstração da realidade.
  - Estruturas de dados e algoritmos estão intimamente ligados.

Introdução

Representação

- Dados podem estar representados (estruturados) de diferentes maneiras.
- Normalmente, a escolha da representação é determinada pelas operações que serão utilizadas sobre eles.
- Exemplo: números inteiros.
  - Representação por palitinhos: $II + IIII = IIIII$.
  - Boa para pequenos números (operação simples).=
  - Representação decimal: $1278 + 321 = 1599$.
  - Boa para números maiores (operação complexa).

Introdução

Exemplo

- Como representar tempo?
- Depende das operações a serem realizadas:
  - *time_t*, tipo inteiro, conta o número de segundos desde 1/1/1970.
  - *struct_timespec*, struct com dois inteiros, um *time_t* e um contador de nanosegundos.
  - *struct_tm*, vários campos para segundo, minuto, dia, mês, ano, horário de verão, etc..
  - *double*, pra armazenar intervalos de tempo.

Introdução

Programa

- Um programa é uma formulação concreta de um algoritmo abstrato, baseado em representações de dados específicas.
- Os programas são feitos em alguma linguagem que pode ser entendida e seguida pelo computador.
- Linguagem de máquina.
- Linguagem de alto nível (uso de compilador).
- Utilizaremos a Linguagem C.

Introdução

Linguagem C

- Criada no início da década de 70 para a programação do sistema operacional Unix
- Uma das linguagens mais utilizadas no mundo, e serviu como base para outras como C++, C#, Java e etc.
- Filosofia: "O programador sabe o que está fazendo".

Introdução

Tipos Abstratos de Dados

- Encapsulam a representação dos dados e as operações que podem ser realizadas sobre eles
- Usuário do TAD *vs.* programador do TAD
  - Usuário só "enxerga" a interface, não a implementação.
  - Não importa se a representação é feita com palitos, números decimais, ou em binário desde que a gente consiga somar, subtrair, multiplicar, etc..
- Os usuários de um TAD só têm acesso às operações disponibilizadas sobre os dados.

Isolamento e Reuso:

- Podemos modificar a implementação do TAD sem modificar o código que usa o TAD.
- Também odemos modificar o código que usa o TAD sem modificar a implementação do TAD.
- O TAD pode ser reaproveitado em vários programas ou módulos.

Implementação:

- Em linguagens orientadas a objeto, como por exemplo C++ e Java, a implementação de TADs é feita por meio de classes.
  - Será estudado na disciplina POO.
- Em linguagem estruturadas, como C e Pascal, a implementação de TADs é feita por meio da definições de tipos e implementação de funções.
- Em C, para definição de TADs, utiliza-se as instruções *typedef* e *struct*.

Introdução

Registros

- Vetores e matrizes
  - Estruturas de dados homogêneas
  - Armazenam vários valores, mas todos de um mesmo tipo (todos int, todos double, todos float, todos char).
- Registro (ou *struct*)
  - Tipo de dado estruturado heterogêneo.
  - Coleção de variáveis referenciadas sobre um mesmo nome.
  - Permite agrupar dados de diferentes tipos numa mesma estrutura (ao contrário de matrizes que possuem elementos de um mesmo tipo).
  - Cada componente de um registro pode ser de um tipo diferente (int, char, ...)

Registros (*struct*):

- Uma estrutura 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 instrução *struct* para representar esse tipo de dado.

Introdução

Registros

Conceito de *struct*:

- Os elementos do registro são chamados de campos ou membros da struct.
- É utilizado para armazenar informações de um mesmo objeto.
- Exemplos:
  - carro $\rightarrow$ cor, marca, ano, placa, chassi.
  - pessoa $\rightarrow$ nome, idade, endereço.

Registros

Exemplo

Exemplo de um registro para armazenar dados de um ponto em duas dimensões:

```
struct ponto {
  float x ;
  float y ;
};
struct ponto p1;
p1.x = 0.0;
p1.y = 0.0;
```

Registros

Exemplo

Exemplo de um registro para armazenar uma fração:

```
struct fracao {
  int numerador;
  int denominador;
};
struct fracao f1;
f1.numerador  = 1;
f1.denominador = 2;
```

Registros

Exemplo

Exemplo de um registro para armazenar dados de um aluno:

```
struct aluno {
  char nome[30];
  int matricula;
};
struct aluno aluno_jose;
strcpy(aluno_jose.nome,"Jose dos Santos");
aluno_jose.matricula = 20169999;
```

Registros

Exemplo

Exemplo de um registro para armazenar uma data:

```
struct data {
  int dia;
  int mes;
  int ano;
};
struct data natal;
natal.dia = 25;
natal.mes = 12;
natal.ano = 2017;
```

Registros

Exemplo

Exemplo de um registro para armazenar dados de um aluno, que é formado por um campo do tipo registro:

```
struct data {
  int dia;
  int mes;
  int ano;
};
struct aluno {
  char nome[30];
  int matricula;
  struct data nasc;
};
struct aluno aluno1;
strcpy(aluno1.nome,"Jose dos Santos");
aluno1.matricula = 20169999;
aluno1.nasc.dia = 1;
```

Declaração de Tipos

Declaração de Tipos

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

Declaração de Tipos

Exemplo

Exemplo da definição do tipo livro:

```
typedef struct {
   char titulo[50];
   char autor[50];
   char assunto[100];
   int codigo;
} livro;
```

Declaração de Tipos

Exemplo

Exemplo de uso do tipo livro:

```
livro book1;
strcpy( book1.titulo, "C Como Programar");
strcpy( book1.autor, "H. M. Deitel");
strcpy( book1.assunto, "Programacao em c");
book1.codigo = 123456;
printf( "Titulo do livro : %s\n", book1.titulo);
printf( "Autor do livro : %s\n", book1.autor);
printf( "Assunto do livro : %s\n", book1.assunto);
printf( "Codigo do livro: %d\n", book1.codigo);
```

Declaração de Tipos

Exemplo

Exemplo de uso de ponteiro com registro:

```
livro *book2;
book2 = malloc(sizeof(livro));
strcpy( book2->titulo, "Projeto de Algoritmos");
strcpy( book2->autor, "Nivio Ziviani");
strcpy( book2->assunto, "Estruturas de dados");
book2->codigo = 999999;
printf( "Titulo do livro : %s\n", book2->titulo);
printf( "Autor do livro : %s\n", book2->autor);
printf( "Assunto do livro : %s\n", book2->assunto);
printf( "Codigo do livro: %d\n", book2->codigo);
```

Tipos Abstratos de Dados

Declaração de Tipos

- 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.

Tipos Abstratos de Dados

Exemplo

TAD para representar números complexos:

```
typedef struct {
  double real;
  double img;
} complexo;
```

Tipos Abstratos de Dados

Exemplo

Função que soma dois números complexos:

```
complexo *soma(complexo *n1, complexo *n2) {
  complexo *soma = malloc(sizeof(complexo));
  soma->real = n1->real + n2->real;
  soma->img = n1->img + n2->img;
  return soma;
}
```

Tipos Abstratos de Dados

Exemplo

Função que multiplica dois números complexos:

```
complexo *multiplica(complexo *n1, complexo *n2) {
  complexo *mul = malloc(sizeof(complexo));
  mul->real = n1->real * n2->real - n1->img * n2->img;
  mul->img = n1->real * n2->img + n1->img * n2->real;
  return mul;
}
```

Tipos Abstratos de Dados

Exemplo

Função que inverte um número complexo:

```
complexo *inverte(complexo *n) {
  complexo *inv = malloc(sizeof(complexo));
  inv->real = (-1) * n->real;
  inv->img = (-1) * n->img;
  return inv;
}
```

Tipos Abstratos de Dados

Exemplo

Funções que inicializa e imprime números complexos:

```
\begin{lstlisting}
void inicializa(complexo *n, double r, double i) {
  n ->real = r;
  n ->img = i;
}

void escreve(complexo *n) {
  printf("%.2f + %.2fi\n", n->real, n->img);
}
```

Tipos Abstratos de Dados

Exercício

Implemente um TAD conta bancária com campos número e saldo que suporte as as seguintes operações:

- Iniciar uma nova conta com um número e saldo.
- Depositar um valor na conta.
- Sacar um valor da conta.
- Imprimir o saldo.

Fim

Algoritmos e Estrutura de Dados II

Tipos Abstratos de Dados

prof. Frederico Santos de Oliveira

Universidade Federal de Mato Grosso
Faculdade de Engenharia