Estrutura de Dados I

Prof. Msc. Luiz Mário Lustosa Pascoal

Qual a diferença entre um algoritmo e um programa?

Algoritmos e Estruturas de Dados

- **Algoritmo:**
  - Sequência de ações executáveis para a solução de um determinado tipo de problema
  - Exemplo: "Receita de Bolo"
- **Programa**
  - Um algoritmo passa a ser um programa após ser convertido para uma linguagem aceita por um computador (Linguagem Binária).
- **Estruturas de Dados**
- Conjunto de dados que representa uma situação real
  - Abstração da realidade
  - Estruturas de Dados e Algoritmos estão intimamente ligados

Representação dos Dados

- Os 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 = IIIIII
    - Boa para pequenos números (operação simples)
  - Representação decimal: 1278 + 321 = 1599
    - Boa para números maiores (operação complexa)

Programas

- 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 / binária (0 ou 1)
  - Linguagem de alto nível (uso de compilador)
    - C, C++, Java, entre outras.

Motivação

- **Por que estudar os tipos de dados?**
- **Benefícios?**
  - Organização da informação
  - Melhora o desempenho
  - Proporciona o reuso de código
  - Proporciona interoperabilidade
  - Diminui custos
- **Necessidade de ED**
  - Ferramenta básica
  - Projetos comerciais
  - Projetos acadêmico
- Duas são as principais preocupações em um projeto de software
  - Os procedimentos a serem executados;
  - Os dados sobre os quais os procedimentos atuam;

Motivação

- **Estrutura de Dados** é um modo particular de **armazenamento** e **organização** dos dados em um computador.
  - Viabiliza o uso de dados de maneira eficiente.
  - Exemplos: Vetores (Arrays), Registros, Listas, Pilhas, Filas, Árvores, Grafos, etc.

Motivação

- As estruturas de dados são chamadas tipos de dados compostos que dividem-se em:
  - **Homogêneos**
    - Conjuntos de dados formados pelo mesmo tipo de dado primitivo.
    - Ex. Vetores e Matrizes.
  - **Heterogêneos**
    - Conjuntos de dados formados por tipos de dados primitivos diferentes em uma mesma estrutura.
    - Ex. Registros e Classes.

Tipos de Dados

- Define a forma como um dado deve ser armazenado ou recuperado, bem como os possíveis valores que ele pode assumir ou as operações que podem ser efetuadas sobre os mesmos
  - Exemplo em Pascal:
    - integer permite valores inteiros e operações de adição, multiplicação, subtração e divisão;
    - string permite valores literais e operações de concatenação;

Tipos de Dados

- Primitivos, derivados ou coleções;
  - Os principais tipos primitivos são: inteiro, real, lógico, caracter, ponteiro;
- Estáticos ou dinâmicos (instanciados em tempo de execução);

Operações

- Um conjunto de instruções a fim de manipular um determinado tipo de dado a fim algum objetivo;
  - Criação (declaração)
  - Percurso
  - Busca
  - Alteração
  - Retirada
  - Inserção (em tipos dinâmicos)

Tipos Primitivos ou Escalares

- Inteiro (integer, longint, etc.);
- Real (real, double, etc.);
- Lógico (boolean);
- Caracter (char);

Tipos Inteiros

- Operações numéricas contidas no conjunto dos números inteiros:
  - Soma, subtração, multiplicação, divisão inteira, resto da divisão;
- Permitem comparações de igualdade e/ou de desigualdade;

Tipos Reais

- Satisfaz as operações e comparações possíveis com tipos inteiros;
- Operações numéricas contidas no conjunto dos números reais:
  - Soma, subtração, multiplicação, divisão;

Tipos Lógicos

- Permite operações lógicas (booleanas):
  - E, OU, NÃO;
- Deve-se ter muito cuidado na construção de expressões lógicas
  - Quanto maiores elas são, maiores as chances de cometermos equívocos.

Tipo Caracter

- Permite a representação de um único caracter;
- Operações de igualdade e desigualdade;
- Por ser armazenado internamente como um valor inteiro, podemos fazer um "casting" e efetuar outras operações.

Tipos Coleções ou Não-Escalares

- Vetor;
- Registro;
- Conjunto.

Tipo Vetor

- Coleção de dados homogênea indexada que pode ser acessada por meio de um índice numérico;
- var v = array [1..5] of integer;
- v[3];

Tipo Registro

- Coleção de dados heterogênea cujas informações podem ser acessadas por meio de um campo;

- var r = record
  c1: integer;
  c2: boolean;
  end;

- r.c1;

Tipo Conjunto

- Coleção de objetos (ou informações) correlatos que podem estar presentes ou não em um dado momento;
- var c = set of (V1, V2, V3);
- c := [V1, V2];
- V1 in c;

Tipos Abstratos de Dados

- Segundo a Wikipédia:
  - Especificação de um conjunto de dados e operações que podem ser executadas sobre esses dados;
- Exemplo:
  - Quando usamos arrays e registros para criar uma estrutura de dados em vez de usar variáveis de tipos primitivos.

Tipos Abstratos de Dados

- Vetores, registros e conjuntos são interessantes...
  - ... Mas há um problema, são estáticos!
- O que acontece se eu tiver um vetor de 5 posições e precisar de outras 1000?
- E se meu vetor tiver 100000 posições e eu somente uso 5, isso é bom?

Alocação de Memória

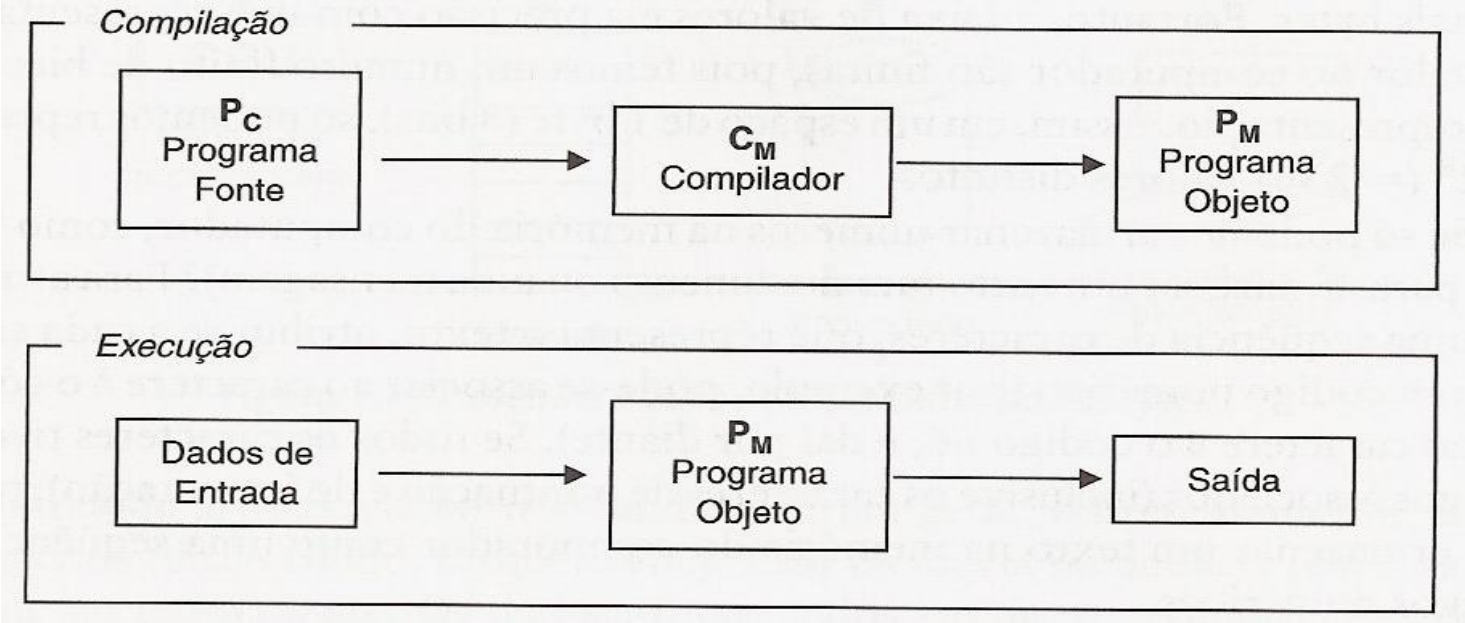
- Alocação estática → Variável alocada ocupa espaço fixo e contíguo na memória;
- Alocação dinâmica → Variável alocada ocupa espaço variável e é criada segundo a necessidade do programa.

Vantagens e Desvantagens da Alocação Dinâmica

- Se alocamos dinamicamente, podemos aumentar e diminuir o tamanho de nossa estrutura quando quisermos!
- Entretanto, necessitaremos de mais algumas operações para buscar, inserir e/ou remover informações;
- Além disso, um array (estático) de vinte posições geralmente ocupa menos espaço que uma lista cujos elementos foram criados um a um dinamicamente.

Compilação de Programas

- Compilação: "tradução" de código fonte (P_C) para linguagem de máquina (M).
- Compilador (C_M), escrito em M: lê o programa P_C e traduz cada instrução para M, escrevendo o programa objeto (P_M)



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

Tipos Abstratos de Dados (TADs)

- Dessa forma, o usuário pode abstrair da implementação específica.
- Qualquer modificação nessa implementação fica restrita ao TAD
- A escolha de uma representação específica é fortemente influenciada pelas operações a serem executadas

Implementação de TADs

- Em linguagens orientadas por 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
- Em C utiliza-se as estruturas de Registro (**Typedef e Structs**).
- Em linguagens orientadas a objetos este conceito pode ser visto através do uso de **classes** (class).

Estruturas (Structs) em C / C++

- Uma estrutura é uma coleção de uma ou mais variáveis, possivelmente de tipos diferentes, 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

Estruturas (Structs) em C / C++

```
#include <iostream>
#include <string>
using namespace std;

struct Aluno {
    string nome;
    int matricula;
    char conceito;
};

main() {
    struct Aluno al, aux;

    al.nome = "Pedro"
    al.matricula = 200712;
    al.conceito = 'A';
    aux = al;
    cout << aux.nome
}
```

al:

Pedro	
200712	A

aux:

Pedro	
200712	A

Declaração de Tipos

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

```
typedef struct {
    string nome;
    int matricula;
    char conceito;
} TipoAluno;


typedef int[10] Vetor;
```

```
int main() {
    TipoAluno al;
    Vetor v;

    ...
}
```