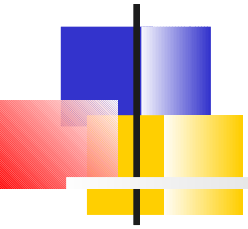
FACCAT
Sistemas de Informação

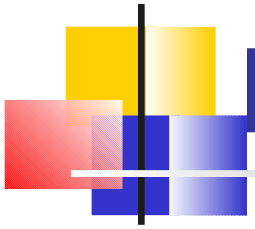


Estruturas de Dados

Aula 1

http://professores.faccat.br/azambuja

(Estrutura de Dados)



Definição de variáveis

- Podemos imaginar uma variável como o local onde se pode colocar qualquer conjunto de valores possíveis do tipo básico associado.

Combinação de Vetores e Registros

Tipo CadAluno: registro

Matr: inteiro;

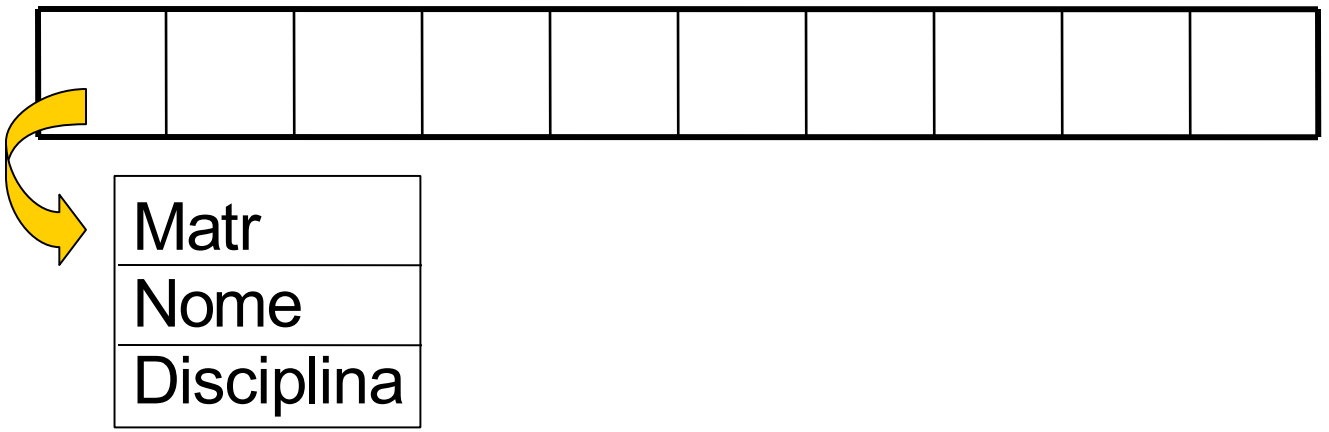
Nome: caracter;

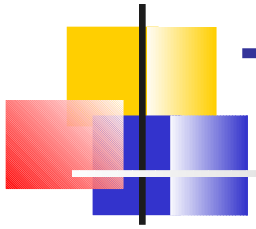
Disciplina: inteiro

Fim_registro;

Tipo Turma: vetor [1:40] CadAluno;

Turma

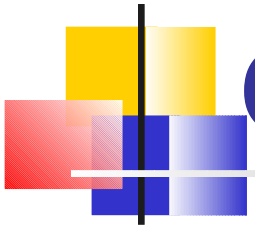




Tipos em Pascal

```
Type CadAluno = record
    Matr: inteiro;
    Nome: caracter;
    Disciplina: inteiro
end;

Turma = array [1:40] of CadAluno;
```



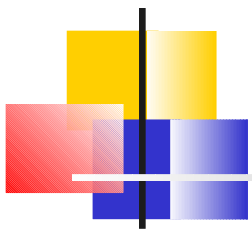
Comandos básicos

- **Comando de Atribuição**

  Para a atribuição de um valor a uma variável, será usado o símbolo de atribuição ←, que tem um caráter imperativo.

  Ex.: soma ← 5 + 4;

  Em Pascal: soma:=5+4;



- **Blocos de comandos básicos de controle**
- Um bloco pode ser definido como um conjunto de comandos com uma função bem definida. Ele serve também para definir os limites onde as variáveis declaradas em seu interior serão conhecidas.
- ✓ **Português Estruturado**

<declaração de variáveis>

Início

<comandos>

fim

- **Seqüência Simples**

É um conjunto de comandos, separados por um ponto e vírgula ‘;’, que serão executados numa seqüência linear de cima para baixo.

Português Estruturado

C_1;

C_2;

C_3;

..

C_n

- **Alternativa**: Quando a ação a ser executada depender de uma inspeção ou teste, têm-se uma alternativa simples ou composta.

- ***Alternativa Simples*** :

Português Estruturado

se <condição>
então
C_1;
C_2;
C_3;
...
C_n;
fim se

- ***Alternativa Composta***:

Português Estruturado

se <condição>

então

C_1;

C_2;

...

C_n;

Senão

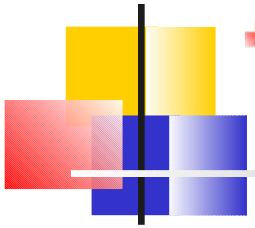
C'_1;

C'_2;

...

C'_n;

fim se



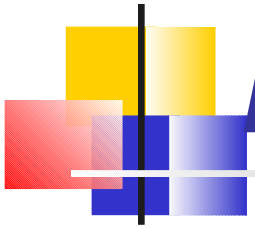
Em Pascal

Simples

If <condição> then <comando>;

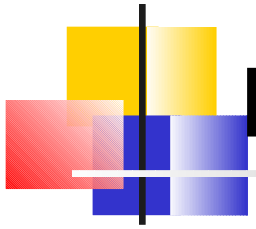
Composta

If <condição> then <comando>
else <comando>;



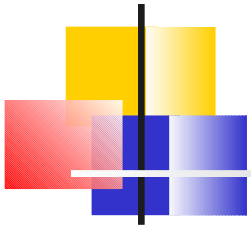
Repetição

- ***Repetição Enquanto***: Quando um conjunto de ações é executado repetidamente enquanto uma determinada condição permanece válida. (Expressão cujo resultado é o valor lógico *verdadeiro*). Também chamada de repetição pré-teste.



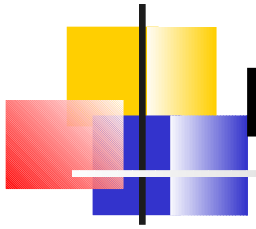
Em Pascal

```
While <condição> do
  begin
      C1;
      C2;
      C3;
      ...
      Cn;
  end;
```



- ***Repetição Repetir***: Quando um conjunto de ações é executado repetidamente até que uma determinada condição se torne válida.

  Também chamada pós-teste.



Em Pascal

repeat

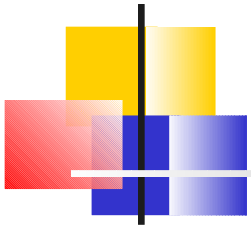
C_1;

C_2;

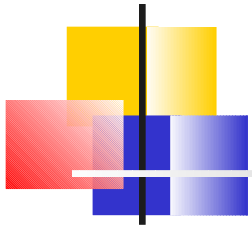
...

C_n;

until <condição>;



- ***Repetição com variável de controle (Loop)***: Neste caso existirá sempre uma inicialização da variável de controle, um teste para verificar se a variável atingiu o limite e um acréscimo na variável de controle. Também chamada de repetição contada.



Em Pascal

For v := i to l do

begin

C_1;

C_2;

...

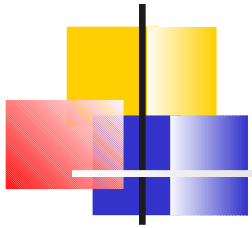
C_n;

end;

Entrada e Saída

Para fornecer os dados ao ambiente exterior ao algoritmo, será necessário imprimi-los ou apresentá-los na tela do computador.

Do mesmo modo, para poder escrever algoritmos mais gerais, será necessário obter dados do ambiente externo para o algoritmo. Os comandos de entrada e saída de dados são os seguintes:



Português Estruturado

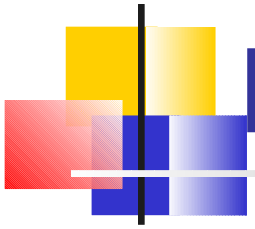
leia (, v_2, v_3);

imprima (v_1, v_2);

Em Pascal

ReadIn (v_1);

WriteIn (v_1);



Modularização

- ***Procedimento***

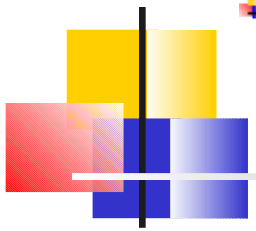
  Permite que um conjunto de comandos usado repetidas vezes em vários algoritmos (ou pontos de um algoritmo) possa ser declarado uma única vez e chamado vários vezes dentro do programa. Os comandos para declaração de procedimentos são os seguintes:

Português Estruturado

Procedimento <nome do procedimento>(lista de parâmetros)
<declaração de variáveis locais>
Início
    <comandos>;
fim

Exemplo

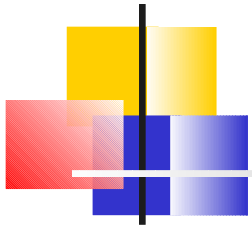
Procedimento Testa (X, Y) // Y é um parâmetro de saída
X, Y: real;
Inicio
    Se X >= 0 então
        Y ← X;
    Senão
        Y ← - X;
    Fim-se
Fim



- ***Função***

Quando ocorre a necessidade de atribuir o resultado da chamada de um procedimento à uma variável, de forma a utilizar este resultado em outros comandos, é necessário que exista um parâmetro de retorno na chamada do procedimento para que este seja usado como resultado da execução do procedimento. Para isto, usa-se um tipo especial de procedimento chamado ***função***. Os comandos para declaração de função são os seguintes:

Português Estruturado



Função <nome da função>(lista de parâmetros): <tipo da variável de retorno>

<declaração de variáveis locais>

Início

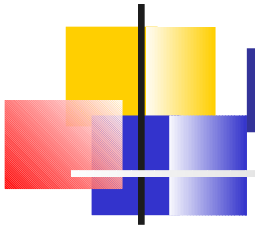
<comandos>

<nome da função> ← <expressão de retorno> //não necessariamente precisa ser no fim

fim

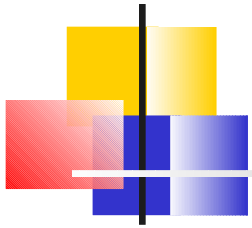
Exemplo

```
Função Testa (X) : real;
X: real;
Inicio
        Se  X >= 0 então
                Testa ← X;
        Senão
                Testa ← - X;
        Fim-se
fim
```

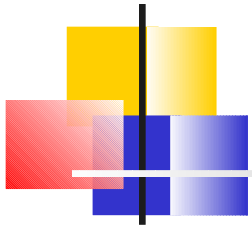


Máximas da programação:

1) Algoritmos devem ser feitos para serem lidos por seres humanos de modo a poder ser compreendido por outras pessoas e serem corrigidos, receber manutennção e serem modificados.

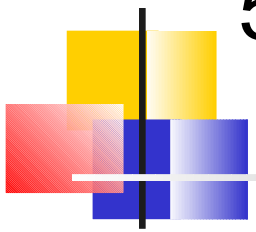


2) Escreva os comentários no momento em que estiver escrevendo o algoritmo. Os comentários devem ser tão bem escritos quanto os algoritmos. A melhor forma de fazê-lo é no momento de maior intimidade com os detalhes, ou seja, durante a resolução do problema.



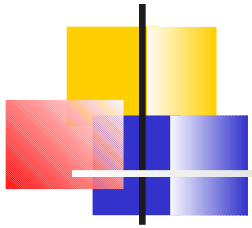
3) Utilize espaço em branco para melhorar a legibilidade.

4) Os comentários deverão acrescentar alguma coisa, dizendo o por quê das estruturas utilizadas.



5) Use comentários no cabeçalho do algoritmo. Contendo, por exemplo:

- Um descrição do que faz o algoritmo
- Como utilizá-lo
- Explicação das variáveis mais importantes
- Estruturas de dados utilizadas
- O nomes de qualquer método especial utilizado, juntamente com referências nas quais mais informações possam ser encontradas
- Autor
- Data de escrita

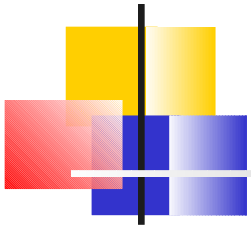


6) Escolha nomes representativos para as variáveis. Uma seleção adequada de nomes de variáveis é o princípio mais importante da legibilidade de algoritmos.

Exemplo: X ← Y + Z

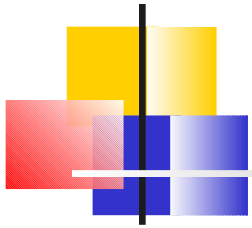
é muito menos claro que

Preço ← Custo + Lucro.



7) Um comando por linha é suficiente.

A utilização de vários comandos por linha é prejudicial por várias razões, dentre as quais destacam-se: dificuldade de leitura do algoritmo e depuração.

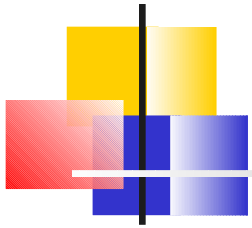


8)Utilize parênteses para aumentar a legibilidade e previnir-se contra erros.

A * B * C / (D * E * F)

Ou

(A * B * C) / (D * E * F)



9) Utilize identação para mostrar e a estrutura lógica do algoritmo.

10) Lembre-se: toda vez que for feita uma modificação no algoritmo, os comentários, associados devem ser alterados e não apenas os comandos.