Problemas Propostos

Problema 1.12.1

Fazer um algoritmo que:

Leia um número indeterminado de linhas contendo cada uma a idade de um indivíduo. A última linha, que não entrará nos cálculos, contem o valor da idade igual a zero. Calcule e escreva a idade media deste grupo de indivíduos

```
algoritmo( )
{
        declare IDADE : inteiro;    // a idade lida de cada individuo
        declare N : inteiro;        // a quantidade de individuos lidos
        declare MEDIA : real;       // a idade media calculada
        declare SOMA : real;        // a soma das idades de cada individuo

        // atribuicao de valores iniciais para o calculo da media

        SOMA := 0;          // acumulador da soma das idades
        N := 0;         // numero de individuos lidos

        Faca
        {
                leia ( "informe a idade: ", IDADE );
                SOMA := SOMA + IDADE;
                N := N + 1;
        } ateque ( IDADE == 0 )

        MEDIA := SOMA / N;
        escreva ( "A idade media e ", MEDIA );
}
```

Problema 1.12.2

Tem-se um conjunto de dados contendo a altura e o sexo (masculino, feminino) de 50 pessoas. Fazer um algoritmo que calcule e escreva:

- a maior e a menor altura do grupo
- a media de altura das mulheres
- numero de homens

```
algoritmo( )
{
        declare MAIOR : real;           // a maior altura do grupo de pessoas
        declare MENOR : real;           // a menor altura do grupo de pessoas
        declare ALTURA: real;           // a altura lida de cada pessoa
        declare SEXO : caracter;        // o sexo (M ou F)
        declare MEDIA : real;           // a media das alturas das mulheres
        declare HOMENS : inteiro;       // o numero de homens
        declare SOMA : real;            // a soma das alturas das mulheres
        declare N : inteiro;            // contador para o numero de pessoas
        declare MULHERES:inteiro;       // numero de mulheres

        // atribuicao de valores iniciais para o calculo da media

        SOMA := 0;          // acumulador da soma das idades das mulheres
        N := 0;             // numero de pessoas lidos
        HOMENS := 0;        // numero de homens
        MULHERES := 0;      // numero de mulheres
        MAIOR := 0;         //
        MENOR := 10;

        enquanto ( N <= 50 )
        {
                leia ( "informe a altura: ", ALTURA );
                leia ( "informe o sexo: ", SEXO );
                se ( ALTURA > MAIOR )
                {
                        MAIOR := ALTURA;
                }
                se ( ALTURA < MENOR )
                {
                        MENOR := ALTURA;
                }
                se ( SEXO == 'M' | SEXO == 'm' )
                {
                        HOMENS := HOMENS + 1;
                }
```

```
            senao
            {
                MULHERES := MULHERES + 1;
                SOMA := SOMA + ALTURA;
            }
            N := N + 1;
        }
        MEDIA := SOMA / MULHERES;
        escreva ( "A maior altura e ", MAIOR );
        escreva ( "A menor altura e ", MENOR );
        escreva ( "A altura media das mulheres e ", MEDIA );
        escreva ( "O numero de homens e ", HOMENS );
}
```

Problema 1.12.3

A conversão de graus Farenheit para centígrados é obtida por

$$C = \frac{5(F - 32)}{9}$$

Fazer um algoritmo que calcule e escreva uma tabela de centígrados em função de graus farenheit, que variam de 50 a 150 de 1 em 1.

```
algoritmo( )
{
        declare C : real;       // graus Centigrados
        declare F : real;       // graus Farenheit

        // atribuicao de valores iniciais

        F := 50;        // valor inicial do intervalo desejado

        repita ateque ( F > 150 )
        {
                C := ( 5 / 9 ) * ( F - 32 );
                escreva ( "Farenheit: ", F, " Centigrados: ", C );
                F := F + 1;
        }
}
```

Problema 1.12.4

Um comerciante deseja fazer o levantamento do lucro das mercadorias que ele comercializa. Para isto, mandou digitar numa linha para cada mercadoria com o nome, preço de compra e preço de venda das mesmas.
Fazer um algoritmo que:

- determine e escreva quantas mercadorias proporcionam:

a) lucro menor que 10%
b) lucro entre 10% e 20%
c) lucro maior que 20%

- determine e escreva o valor total de compra e de venda de todas as mercadorias, assim como o lucro total.

Obs: o aluno deve adotar um flag.

```
algoritmo( )
{
        declare NOME : cadeia;              // nome da mercadoria
        declare PRECO_COMPRA : real;        // preco de compra da mercadoria
        declare PRECO_VENDA : real;         // preco de venda da mercadoria
        declare LUCRO : real;               // lucro calculado de cada mercadoria
        declare QUANT1 : inteiro;           // qtd de mercadorias com lucro ate 10%
        declare QUANT2 : inteiro;           // qtd de mercadorias com lucro entre 10 e
20%
        declare QUANT3 : inteiro;           // qtd de mercadorias com lucro > 20%
        declare TOTAL_COMPRA : real;              // valor total da compra
        declare TOTAL_VENDA : real;               // valor total da venda

        // inicializacao dos acumuladores

        TOTAL_COMPRA := 0;
        TOTAL_VENDA := 0;
        QUANT1 := 0;
        QUANT2 := 0;
        QUANT3 := 0;

        repita ateque ( NOME == "FIM" )
        {
                leia ( "informe o nome da mercadoria: ", NOME );

                se ( NOME <> "FIM" )
                {
                        leia ( "informe o preco de compra: ", PRECO_COMPRA );
                        leia ( "informe o preco de venda: ", PRECO_VENDA );
                        LUCRO := 100 * (PRECO_VENDA - PRECO_COMPRA) /
                                        PRECO_COMPRA;
                        se ( LUCRO < 10 )
                        {
```

```
                        QUANT1 := QUANT1 + 1;
                }
                senao
                {
                        se ( LUCRO >= 10 & LUCRO <= 20 )
                        {
                                QUANT2 := QUANT2 + 1;
                        }
                        senao
                        {
                                QUANT3 := QUANT3 + 1;
                        }
                }
                TOTAL_COMPRA := TOTAL_COMPRA + PRECO_COMPRA;
                TOTAL_VENDA := TOTAL_VENDA + PRECO_VENDA;
        }
        LUCRO := 100 * ( TOTAL_VENDA - TOTAL_COMPRA ) /
                        TOTAL_COMPRA;
    }
    escreva ( "Quantidade de mercadorias com lucro < 10%: ", QUANT1 );
    escreva ( "Quantidade de mercadorias com lucro < 20%: ", QUANT2 );
    escreva ( "Quantidade de mercadorias com lucro > 20%: ", QUANT3 );
    escreva ( "Valor total das compras: ", TOTAL_COMPRA );
    escreva ( "Valor total das vendas: ", TOTAL_VENDA );
    escreva ( "Lucro total (%): ", LUCRO );
}
```

Problema 1.12.5

Supondo que a população de um pais A seja da ordem de 90.000.000 de habitantes com uma taxa anual de crescimento de 3% e que a população de um pais B seja, aproximadamente, de 200.000.000 de habitantes com uma taxa anual de crescimento de 1,5%, fazer um algoritmo que calcule e escreva o numero de anos necessários para que a população do pais A ultrapasse ou iguale a população do pais B, mantidas essas taxas de crescimento

```
algoritmo( )
{
        // declaracao das variaveis

        declare popA : real;          // populacao do pais A
        declare popB : real;          // populacao do pais B
        declare taxaA : real;         // taxa de crescimento da populacao de A
        declare taxaB : real;         // taxa de crescimento da populacao de B
        declare anos : inteiro;       // numero de anos para que popA >= popB

        // inicializacao dos acumuladores e constantes

        popA := 90000;
        popB := 200000;
        taxaA := 0.03;
        taxaB := 0.015;
        anos := 0;

        repita ateque ( popA >= popB )
        {
                popA := popA + ( popA * taxaA );
                popB := popB + ( popB * taxaB );
                anos := anos + 1;
        }
        escreva ( "Tempo decorrido em anos: ", anos );
        escreva ( "Populacao do pais A: ", popA );
        escreva ( "Populacao do pais B: ", popB );
}
```

Problema 1.12.6

Um determinado material radioativo perde metade se sua massa a cada 50 segundos. Dada a massa inicial, em gramas, fazer um algoritmo que determine o tempo necessário para que essa massa se torne menor do que 0,5 grama. Escreva a massa inicial, a massa final e o tempo calculado em horas, minutos e segundos.

```
algoritmo()
{
	// declaracao das variaveis

	declare massaI : real;		// massa inicial do material radioativo
	declare massaF : real;		// massa final do material radioativo
	declare tempo : real;		// tempo em segundos
	declare horas : inteiro;	// as horas de tempo
	declare minutos : inteiro;	// os minutos de tempo
	declare segundos : inteiro;	// os segundos de tempo

	// inicializacao dos acumuladores e constantes

	tempo := 0;
	horas := 0;
	minutos := 0;
	segundos := 0;

	leia ( "informe a massa inicial (em gramas): ", massaI );
	massaF := massaI;

	repita ateque ( massaF < 0.5 )
	{
		massaF := massaF / 2;
		tempo := tempo + 50;
	}

	// transformando o tempo em horas, minutos e segundos

	segundos := tempo;

	enquanto ( segundos >= 60 )
	{
		minutos := minutos + 1;
		se ( minutos == 60 )
		{
			horas := horas + 1;
			minutos := 0;
```

```
                }
                segundos := segundos - 60;
        }

        // escrevendo os resultados

        escreva ( "Massa inicial (em gramas): ", massaI );
        escreva ( "Massa final (em gramas): ", massaF );
        escreva ( "Tempo decorrido em segundos: ", tempo );
        escreva ( horas, " horas, ", minutos, "minutos e ",segundos, " segundos" );
}
```

Problema 1.12.7

Deseja-se fazer um levantamento a respeito da ausência de alunos a primeira prova de programação de computadores para cada uma das 14 turmas existentes. Para cada turma é fornecido um conjunto de valores, sendo que os dois primeiros valores do conjunto corresponde a identificação da turma (A, B, C ...) e ao número de alunos matriculados, e os demais valores deste conjunto contem o número de matricula do aluno e a letra A ou P para o caso de o aluno estar ausente ou presente, respectivamente. Fazer um algoritmo que:
a) para cada turma, calcule a porcentagem de ausência e escreva a identificado da turma e a porcentagem calculada.
b) determine e escreva quantas turmas tiveram porcentagem de ausência superior a 5%.

```
algoritmo( )
{
        // declaracao das variaveis

        declare TURMA : caracter;       // identificacao da turma
        declare ALUNOS : inteiro;       // numero de alunos matriculados na turma
        declare MATRICULA : cadeia;     // numero de matricula do aluno
        declare CHAMADA : caracter;     // A ou P (ausente ou presente)
        declare QUANT_A : inteiro;      // quantidade de alunos ausentes por turma
        declare QUANT_P : inteiro;      // qtidade de alunos presentes por turma
        declare PORCENT : real;         // porcentagem de ausencia por turma
        declare N_TUR: inteiro;         // contador para o numero de turmas
        declare N_ALU : inteiro;        // cont para o numero de alunos por turma
        declare T : inteiro;            // contador para turmas com ausencia > 5%

        // inicializacao dos acumuladores gerais

        N_TUR := 0;         // contador ate 14 turmas
        T := 0;             // total de turmas com ausencia maior que 5%

        repita ateque ( N_TUR == 14 )
        {
                // inicializacao dos acmuladores por turmas

                QUANT_A := 0;
                QUANT_P := 0;
                N_ALU := 0;         // contador ate o numero de alunos da turma
                leia ( "informe a turma: ", TURMA );
                leia ( "informe o numero de alunos matriculados: ", ALUNOS );
                repita ateque ( N_ALU == ALUNOS )
                {
                        leia ( "informe o numero de matricula: ", MATRICULA );
                        leia ( "Chamada (P/A): ", CHAMADA );
```

```
                    se ( CHAMADA == 'P' )
                    {
                        QUANT_P := QUANT_P + 1;
                    }
                    senao
                    {
                        se ( CHAMADA == 'A' )
                        {
                            QUANT_A := QUANT_A + 1;
                        }
                    }
                    N_ALU := N_ALU + 1;
                }
                PORCENT := 100 * ( QUANT_A / N_ALU );
                escreva ( "Turma: ", TURMA, "% faltas: ", PORCENT );
                se ( PORCENT > 5 )
                {
                    T = T + 1;
                }
                N_TUR := N_TUR + 1;        // proxima turma
            }
            escreva ( "Numero de turmas com ausencia maior que 5%: ", T );
}
```

Problema 1.12.8

Uma certa firma fez uma pesquisa de mercado para saber se as pessoas gostaram ou não de um novo produto lançado no mercado. Para isso, forneceu o sexo do entrevistado e sua resposta (sim ou não). Sabendo-se que foram entrevistados 2.000 pessoas, fazer um algoritmo que calcule e escreva:
a) o número de pessoas que responderam sim
b) o número de pessoas que responderam não
c) a porcentagem de pessoas do sexo feminino que responderam sim
d) a porcentagem de pessoas do sexo masculino que responderam não

```
algoritmo( )
{
        // declaracao das variaveis

        declare HOMENS : inteiro;          // numero de homens entrevistados
        declare MULHERES : inteiro;        // numero de mulheres entrevistadas
        declare R_SIM : inteiro;           // numero de pessoas que responderam sim
        declare R_NAO : inteiro;           // numero de pessoas que responderam não
        declare RESP : caracter;           // resposta sim ou nao (S ou N, s ou n)
        declare SEXO : caracter;           // sexo do entrevistado (M ou F, m ou f)
        declare R_SIM_M : inteiro;         // contador de respostas sim de mulher
        declare R_NAO_H : inteiro;         // contador de respostas nao de homem
        declare N : inteiro;               // contador para o numero de pessoas
        declare P_SIM_M : real;            // porcentagem de respostas sim de mulher
        declare P_NAO_H : real;            // porcentagem de respostas nao de homem
        declare ENTREVISTADOS : inteiro;

        // inicializacao dos acumuladores gerais

        N := 0;/* contador ate 2000 pessoas */
        HOMENS := 0;
        MULHERES := 0;
        R_SIM := 0;
        R_NAO := 0;
        R_SIM_M := 0;
        R_NAO_H := 0;
        ENTREVISTADOS := 4;

        repita ateque ( N == ENTREVISTADOS )
        {
                leia ( "informe o sexo: ", SEXO );
                leia ( "gostou do produto (S/N): ", RESP );

                // verifica se a pessoa e homem ou mulher
```

```
        se ( SEXO == 'M' | SEXO == 'm' )
        {
                HOMENS := HOMENS + 1;
                se ( RESP == 'N' | RESP == 'n' )
                {
                        R_NAO_H := R_NAO_H + 1;
                }
        }
        senao
        {
                se ( SEXO == 'F' | SEXO == 'f' )
                {
                        MULHERES := MULHERES + 1;
                        se ( RESP == 'S' | RESP == 's' )
                        {
                                R_SIM_M := R_SIM_M + 1;
                        }
                }
        }

        // verifica se a resposta da pessoa e sim ou não

        se ( RESP == 'S' | RESP == 's' )
        {
                R_SIM := R_SIM + 1;
        }
        senao
        {
                se ( RESP == 'N' | RESP == 'n' )
                {
                        R_NAO := R_NAO + 1;
                }
        }
        N := N + 1;
}

// calculo dos porcentuais

P_SIM_M := 100 * ( R_SIM_M / ENTREVISTADOS );
P_NAO_H := 100 * ( R_NAO_H / ENTREVISTADOS );

// resultados

escreva ( "quantidade de pessoas que responderam sim: ", R_SIM );
escreva ( "quantidade de pessoas que responderam nao: ", R_NAO );
```

```
        escreva ( "% de mulheres que responderam sim: ", P_SIM_M );
        escreva ( "% de homens que responderam nao: ", P_NAO_H );
}
```

Problema 1.12.9

Foi feita uma pesquisa para determinar o índice de mortalidade infantil em certo período. Fazer um algoritmo que:

- leia inicialmente o número de crianças nascidas no período
- leia, em seguida, um numero indeterminado de linhas, contendo, cada uma, o sexo de uma criança morta (masculino, feminino) e o numero de meses de vida da criança. A última linha, que não entrará nos cálculos contem no lugar do sexo a palavra "vazio".

Determine e imprima:
a) a porcentagem de crianças mortas no período
b) a porcentagem de crianças do sexo masculino mortas no período
c) a porcentagem de crianças que viveram 24 meses ou menos no período

```
algoritmo()
{
        // declaracao das variaveis

        declare NASCIDAS : inteiro;      // numero de criancas nascidas no periodo
        declare SEXO : cadeia;           // sexo da crianca
        declare IDADE : inteiro;         // numero de meses de vida da crianca
        declare P_MORTOS : real;         // % de criancas mortas no periodo
        declare P_MORTOS_M : real;       // % de criancas mortas no periodo do sexo
                                         // masculino
        declare P_24MESES : real;        // % de criancas que viveram 24 meses
        declare N_MORTOS : inteiro;      // numero de criancas mortas no periodo
        declare N_MORTOS_M : inteiro;
                // numero de criancas mortas no periodo do sexo masculino
        declare N_24MESES : inteiro;
                // numero de criancas que viveram 24 meses ou menos

        // inicializacao dos acumuladores e contadores

        N_MORTOS := 0;
        N_MORTOS_M := 0;
        N_24MESES := 0;

        leia ( "informe o numero de criancas nascidas: ", NASCIDAS );
        repita ateque ( SEXO == "vazio" )
        {
                leia ( "informe o sexo da crianca: ", SEXO );
                se ( SEXO <> "vazio" )
                {
                        leia ( "informe o tempo de vida (em meses): ", IDADE );
                        N_MORTOS := N_MORTOS + 1;

                        // criancas do sexo masculino
```

```
                    se ( SEXO == "masculino" )
                    {
                        N_MORTOS_M := N_MORTOS_M + 1;
                    }

                    // criancas que viveram 24 meses ou menos

                    se ( IDADE <= 24 )
                    {
                        N_24MESES := N_24MESES + 1;
                    }
                }
            }

            // calculo das porcentagens

            P_MORTOS := 100 * ( N_MORTOS / NASCIDAS );
            P_MORTOS_M := 100 * ( N_MORTOS_M / NASCIDAS );
            P_24MESES := 100 * ( N_24MESES / NASCIDAS );

            // saida dos resultados

            escreva ( "% de criancas mortas no periodo: ", P_MORTOS );
            escreva ( "% de criancas mortas no periodo do sexo masculino: ",
P_MORTOS_M);
            escreva ( "% de criancas que viveram ate 24 meses: ", P_24MESES );
}
```

Problema 1.12.10

Foi feita uma pesquisa de audiência de canal de TV em várias casas de uma certa cidade, num determinado dia. Para cada casa visitada, é fornecido o número do canal (4,5,7,12) e o numero de pessoas que o estavam assistindo naquela casa. Se a televisão estivesse desligada, nada era anotado, ou seja, esta casa não entrava na pesquisa. Fazer um algoritmo que:

- leia um numero indeterminado de dados, sendo que o flag corresponde ao número de canal igual a zero.
- calcule a porcentagem de audiência para cada emissora
- escreva o numero do canal e a sua respectiva porcentagem

```
algoritmo()
{
        // declaracao das variaveis

        declare CANAL : inteiro;            // numero do canal
        declare ESPECTADORES : inteiro;
                // numero de pessoas que estavam assistindo
        declare P_CANAL4 : real;            // porcentagem de audiencia do canal 4
        declare P_CANAL5 : real;            // porcentagem de audiencia do canal 5
        declare P_CANAL7 : real;            // porcentagem de audiencia do canal 7
        declare P_CANAL12 : real;           // porcentagem de audiencia do canal 12
        declare N_CANAL4 : inteiro;         // numero de pessoas vendo o canal 4
        declare N_CANAL5 : inteiro;         // numero de pessoas vendo o canal 5
        declare N_CANAL7 : inteiro;         // numero de pessoas vendo o canal 7
        declare N_CANAL12 : inteiro;        // numero de pessoas vendo o canal 12
        declare PESSOAS : inteiro;          // total de pessoas

        // inicializacao dos acumuladores e contadores

        N_CANAL4 := 0;
        N_CANAL5 := 0;
        N_CANAL7 := 0;
        N_CANAL12 := 0;
        PESSOAS := 0;
        CANAL := 1;             // apenas para forcar a entrada no laco pela 1ª vez

        repita ateque ( CANAL == 0 )
        {
                leia ( "informe o canal que estava sendo assistido: ", CANAL );
                se ( CANAL <> 0 )
                {
                        leia ( "quantas pessoas estavam assistindo: ", ESPECTADORES );
                        PESSOAS := PESSOAS + ESPECTADORES;
                        se ( CANAL == 4 )
```

```
                    {
                        N_CANAL4 := N_CANAL4 + ESPECTADORES;
                    }
                    senao se ( CANAL == 5 )
                    {
                        N_CANAL5 := N_CANAL5 + ESPECTADORES;
                    }
                    senao se ( CANAL == 7 )
                    {
                        N_CANAL7 := N_CANAL7 + ESPECTADORES;
                    }
                    senao se ( CANAL == 12 )
                    {
                        N_CANAL12 := N_CANAL12 + ESPECTADORES;
                    }
                }
        }

        // calculo das porcentagens das emissoras

        P_CANAL4 := 100 * ( N_CANAL4 / PESSOAS );
        P_CANAL5 := 100 * ( N_CANAL5 / PESSOAS );
        P_CANAL7 := 100 * ( N_CANAL7 / PESSOAS );
        P_CANAL12 := 100 * ( N_CANAL12 / PESSOAS );

        // resultados

        escreva ( "% canal 4: ",  P_CANAL4 );
        escreva ( "% canal 5: ",  P_CANAL5 );
        escreva ( "% canal 7: ",  P_CANAL7 );
        escreva ( "% canal 12: ", P_CANAL12 );
}
```

Problema 1.12.11

Uma universidade deseja fazer um levantamento a respeito de seu concurso vestibular. Para cada curso é fornecido o seguinte conjunto de valores.

- código do curso
- numero de vagas
- numero de candidatos do sexo masculino
- numero de candidatos do sexo feminino

O último conjunto, para indicar fim de dados, contem o código do curso igual a zero. Fazer um algoritmo que:

- calcule e escreva, para cada curso, o número de candidatos por vaga a porcentagem de candidatos do sexo feminino (escreva também o código correspondente do curso)
- determine o maior numero de candidatos por vaga e escreva esse numero juntamente com o código do curso correspondente (supor que não haja empate)
- calcule e escreva o total de candidatos

```
algoritmo()
{
        // declaracao das variaveis

        declare CURSO: inteiro;             // codigo do curso
        declare VAGAS : inteiro;            // numero de vagas para cada curso
        declare MASC: inteiro;              // numero de candidatos do sexo masculino
        declare FEMI: inteiro;              // numero de candidatos do sexo feminino
        declare POR_VAGA: real;             // numero de candidatos por vaga por curso
        declare POR_CURSO: real;            // numero de candidatos por curso
        declare PC_FEMI : real;             // % de candidatos do sexo feminino
        declare MAIOR_POR_VAGA : real;
                // maior numero de candidatos por vaga
        declare CURSO_MAIOR : inteiro; // numero do curso de MAIOR_POR_VAGA
        declare TOTAL_CAND : inteiro;   // numero total de candidatos

        // inicializacao dos acumuladores

        TOTAL_CAND := 0;          // acumulador do total de candidatos

        // inicializacao das condicoes iniciais

        MAIOR_POR_VAGA := 0;          // valor inicial para 1ª comparacao
        CURSO := 1;                   // apenas para forcar a entrada no laco a 1ª vez

        repita ateque ( CURSO == 0 )
        {
                leia ( "informe o codigo do curso: ", CURSO );

                se ( CURSO <> 0 )
```

```
            {
                leia ( "numero de vagas: ", VAGAS );
                leia ( "numero de candidatos do sexo masculino: ", MASC );
                leia ( "numero de candidatos do sexo feminino: ", FEMI );
                POR_CURSO := MASC + FEMI;
                POR_VAGA := POR_CURSO / VAGAS;
                PC_FEMI := 100 * ( FEMI / POR_CURSO );
                se ( POR_VAGA > MAIOR_POR_VAGA )
                {
                    MAIOR_POR_VAGA := POR_VAGA;
                    CURSO_MAIOR := CURSO;
                }
                TOTAL_CAND := TOTAL_CAND + POR_CURSO;
                escreva ( "curso: ", CURSO );
                escreva ( "candidatos / vaga: ", POR_VAGA );
                escreva ( "% feminino: ", PC_FEMI );
            }
        }
        escreva ( "total de candidatos: ", TOTAL_CAND );
}
```

Problema 1.12.12

O sistema de avaliação de uma determinada disciplina obedece aos seguintes critérios:

- durante o semestre são dadas três notas
- a nota final e obtida pela media aritmética das notas dadas durante o curso
- é considerado aprovado o aluno que obtiver a nota final superior ou igual a 60 e que tiver comparecido a um mínimo de 40 aulas.

Fazer um algoritmo que:

a) leia um conjunto de dados contendo o numero de matrícula, as três notas e a frequência (número de aulas frequentadas) de 100 alunos.
b) calcule
- a nota final de cada aluno
- a maior e a menor nota da turma
- a nota media da turma
- total de alunos reprovados
- a porcentagem de alunos reprovados por frequencia

c) escreva
- para cada aluno, o numero de matricula, a frequencia, a nota final e o código (aprovado ou reprovado)
- todos os itens que foram calculados na letra b.

```
algoritmo()
{
        // declaracao das variaveis

        declare MATRICULA: inteiro;     // o numero de matricula do aluno
        declare N1,N2,N3,NF: real;      // cada uma das notas do aluno
        declare FREQUENCIA: inteiro;    // frequencia do aluno em numero de aulas
        declare MAIOR: real;            // maior nota da turma
        declare MENOR: real;            // menor nota da turma
        declare MEDIA: real;            // nota media da turma
        declare REPROVADOS: inteiro;    // total de alunos reprovados
        declare REPRO_FRE: inteiro;     // total de alunos reprovados por
frequencia
        declare ALUNOS: inteiro;        // total de alunos
        declare PC_REPRO_FRE: real;
                // porcentagem de reprovados por frequencia
        declare RESULTADO: cadeia;      // "aprovado" ou "reprovado"

        // inicializacao dos acumuladores

        ALUNOS := 0;            // acumulador do total de alunos
        REPRO_FRE := 0;         // acumulador de alunos reprovados por frequencia
        MEDIA := 0;
        REPROVADOS := 0;
```

```
// inicializacao das condicoes iniciais

MAIOR := -1;        // valor inicial para 1ª comparacao, todas as notas serao
                    // maiores que 0 (o valor inicial deve ser pelo menos 1
                    // abaixo do limite inferior)

MENOR := 110;       // valor inicial para 1ª comparacao, todas as notas serão
                    // menores que 100 (o valor inicial deve pelo 1 acima do
                    // limite superior

repita ateque ( ALUNOS == 100 )
{
        leia ( "informe o numero de matricula: ", MATRICULA );
        leia ( "informe a Nota 1: ", N1 );
        leia ( "informe a Nota 2: ", N2 );
        leia ( "informe a Nota 3: ", N3 );
        leia ( "informe a Frequencia: ", FREQUENCIA );

        NF := (N1 + N2 + N3) / 3;

        se ( NF > MAIOR )
        {
                MAIOR := NF;
        }
        se ( NF < MENOR )
        {
                MENOR := NF;
        }

        MEDIA := MEDIA + NF;

        // verificacao da aprovacao ou reprovacao

        se ( NF >= 60 & FREQUENCIA >= 40 )
        {
                RESULTADO := "APROVADO";
        }
        senao
        {
                RESULTADO := "REPROVADO";
                REPROVADOS := REPROVADOS + 1;
                se ( FREQUENCIA < 40 )
                {
                        REPRO_FRE := REPRO_FRE + 1;
                }
```

```
            }

            // saida dos dados de cada aluno

            escreva ( "Numero de matricula: ", MATRICULA );
            escreva ( "Frequencia: ", FREQUENCIA );
            escreva ( "Nota final: ", NF );
            escreva ( "Resultado: ", RESULTADO );

            ALUNOS := ALUNOS + 1;
        }

        // calculo da nota media da turma

        MEDIA := MEDIA / ALUNOS;

        // calculo da porcentagem de alunos reprovados por frequencia

        PC_REPRO_FRE := REPRO_FRE / ALUNOS;
                        /* ou := REPRO_FRE / REPROVADOS */

        // saida dos resultados finais

        escreva ( "Maior nota: ", MAIOR );
        escreva ( "Menor nota: ", MENOR );
        escreva ( "Media da turma: ", MEDIA );
        escreva ( "Total de alunos reprovados: ", REPROVADOS );
        escreva ( "% de reprovados por frequencia: ", PC_REPRO_FRE );
}
```

Problema 1.12.13

Deseja-se fazer uma pesquisa a respeito do consumo mensal de energia elétrica em uma determinada cidade. Para isso, são fornecidos os seguintes dados:

- preço do kwh consumido
- numero do consumidor
- quantidade de kwh consumido durante o mês
- código do tipo de consumidor (residencial, comercial, industrial).
- numero do consumidor igual a zero deve ser usado como flag.

Fazer um algoritmo que:

- leia os dados descritos acima e
- calcule
  a) para cada consumidor o total a pagar
  b) o maior consumo verificado
  c) o menor consumo verificado
  d) o total do consumo para cada um dos tres tipos de consumidores
  e) a media geral de consumo
- escreva
  a) para cada consumidor, o total a pagar
  b) o que foi calculado nos itens acima

```
algoritmo()
{
        // declaracao das variaveis

        declare CONSUMIDOR: inteiro;   // numero de identificacao do consumidor
        declare PRECO: real;           // preco do kwh
        declare CONSUMO: real;         // consumo em kwh no mês
        declare TIPO: caracter;        // tipo de consumidor (c,r,i)
        declare MAIOR: real;           // maior consumo verificado
        declare MENOR: real;           // menor consumo verificado
        declare MEDIA: real;           // media geral de consumo
        declare TOTAL_R: real;         // total de consumo para tipo residencial
        declare TOTAL_C: real;         // total de consumo para tipo comercial
        declare TOTAL_I: real;         // total de consumo para tipo industrial
        declare N: inteiro;            // numero de consumidores
        declare A_PAGAR: real;         // total a pagar pelo consumidor

        // inicializacao dos acumuladores

        TOTAL_R := 0;       // acumulador do total para tipo residencial
        TOTAL_C := 0;       // acumulador do total para tipo comercial
        TOTAL_I := 0;       // acumulador do total para tipo industrial
        N := 0;

        // inicializacao das condicoes iniciais
```

```
MAIOR := -1;          /* o valor inicial deve ser abaixo do limite inferior) */
MENOR := 1000;        /* o valor inicial deve ser acima do limite superior) */

// incializacao das condicoes de contorno

CONSUMIDOR := 1;          // apenas para forcar a entrada inicial no laco

repita ateque ( CONSUMIDOR == 0 )
{
        leia ( "informe o numero do consumidor: ", CONSUMIDOR );
        se ( CONSUMIDOR == 0 )
        {
                interrompa;
        }
        leia ( "informe o preco do kwh: ", PRECO );
        leia ( "informe o consumo mensal: ", CONSUMO );
        leia ( "informe o tipo de consumidor (r,c,i): ", TIPO );

        A_PAGAR := CONSUMO * PRECO;

        // descobre valores maximos e minimos

        se ( CONSUMO > MAIOR )
        {
                MAIOR := CONSUMO;
        }
        se ( CONSUMO < MENOR )
        {
                MENOR := CONSUMO;
        }

        // verificacao do tipo de consumidor

        selecao
        {
                caso ( TIPO == 'R' | TIPO == 'r' )
                {
                        TOTAL_R := TOTAL_R + CONSUMO;
                }
                caso ( TIPO == 'C' | TIPO == 'c' )
                {
                        TOTAL_C := TOTAL_C + CONSUMO;
                }
                caso ( TIPO == 'I' | TIPO == 'i' )
                {
```

```
                        TOTAL_I := TOTAL_I + CONSUMO;
                    }
                }

                // saida dos dados de cada aluno

                escreva ( "Numero do consumidor: ", CONSUMIDOR );
                escreva ( "Total a pagar: ", A_PAGAR );

                N := N + 1;
        }

        // calculo do consumo medio geral

        MEDIA := (TOTAL_R + TOTAL_C + TOTAL_I) / N;

        // saida dos resultados finais

        escreva ( "Maior consumo: ", MAIOR );
        escreva ( "Menor consumo: ", MENOR );
        escreva ( "Media de consumo: ", MEDIA );
        escreva ( "Total residencial: ", TOTAL_R );
        escreva ( "Total comercial: ", TOTAL_C );
        escreva ( "Total industrial: ", TOTAL_I );
}
```

Problema 1.12.14

Tem-se uma estrada ligando várias cidades. Cada cidade tem seu marco quilométrico. Fazer um algoritmo que:

- leia vários pares de dados, contendo cada par os valores dos marcos quilométricos, em ordem crescente, de duas cidades. O último par contem estes dois valores iguais
- calcule os tempos decorridos para percorrer a distancia entre duas cidades com as seguintes velocidades: 20, 30, 40, 50, 60, 70 e 80 km/h, sabendo-se que t = e/v, onde t = tempo, e = espaço e v = velocidade
- escreva os marcos quilométricos, a velocidade e o tempo decorrido entre as duas velocidades, apenas quando este tempo for superior a 2 horas.

```
algoritmo( )
{
        // declaracao das variaveis

        declare T: real;            // tempo em horas entre duas cidades
        declare E: real;            // distancia em km entre duas cidades
        declare V: real;            // velocidade de percurso entre as duas cidades
        declare KM1: real;          // marco quilometrico 1
        declare KM2: real;          // marco quilometrico 2

        // incializacao das condicoes de contorno
        // apenas para forcar a entrada inicial no laco

        KM1 := 0;
        KM2 := 1;

        repita ateque ( KM1 == KM2 )
        {
                // leitura dos marcos quilometricos

                leia ( "informe o marco quilometrico 1: ", KM1 );
                leia ( "informe o marco quilometrico 2: ", KM2 );

                se ( KM1 == KM2 )
                {
                        interrompa;
                }

                // calculo da distancia entre as duas cidades (ou os dois marcos)
                E := KM2 - KM1;

                // calculo do tempo de percurso entre os dois marcos
                // com as varias velocidades
```

```
            V := 20;

            repita ateque ( V > 80 )
            {
                  T := E / V;
                  se ( T > 2 )
                  {
                        escreva ( "Entre ", KM1, " e ", KM2,
                  " a ", V, " km/h", " gastou ", T, " horas" );
                  }
                  V := V + 10;
            }
      }
}
```

Problema 1.12.15

Os bancos atualizam diariamente as contas de seus clientes. Essa atualização envolve a analise dos depósitos e retiradas de cada conta. Numa conta de balanço mínimo, uma taxa de serviço é deduzida se a conta cai abaixo de uma certa quantia especificada. Suponha que uma conta particular comece o dia com um balanço de R$ 60,00. O balanço mínimo exigido é R$ 30,00 e se o balanço de fim de dia dor menor do que isso, uma taxa e reduzida da conta. A fim de que essa atualização fosse feita utilizando computador, e fornecido o seguinte conjunto de dados:

- a primeira linha contem o valor do balanço mínimo diário, quantidade de transações e taxa de serviço
- as linhas seguintes contem numero da conta, valor da transação e o código da transação (depósito ou retirada)

Escrever um algoritmo que:

- calcule o balanço (saldo/débito) da conta do dia
- escreva, para cada conta, o seu numero e o balanço calculado. Se não houver fundos, imprima o numero da conta e a mensagem "NAO HA FUNDOS".

```
algoritmo()
{
        // declaracao das variaveis

        declare MINIMO: real;           // balanco minimo diario
        declare TRANS: inteiro;         // quantidade de transacoes do dia
        declare TAXA: real;             // valor da taxa de servico
        declare CONTA: inteiro;         // numero da conta
        declare VALOR: real;            // valor da transacao
        declare TIPO: caracter;         // tipo transacao (d-deposito, r-retirada)
        declare SALDO: real;            // saldo da conta

        SALDO := 0;

        // leitura das condicoes do banco

        leia ( "informe o valor do balanco minimo diario: ", MINIMO );
        leia ( "informe a quantidade de transacoes: ", TRANS );
        leia ( "informe o valor da taxa de servico %: ", TAXA );

        repita ateque ( TRANS == 0 )
        {
                // leitura dos valores das contas

                leia ( "informe o numero da conta: ", CONTA );
                leia ( "informe o valor da transacao: ", VALOR );
                leia ( "informe o tipo da transacao (d/r): ", TIPO );

                // verifica se e uma operacao de deposito ou retirada
```

```
            se ( TIPO == 'd' | TIPO == 'D' )
            {
                  SALDO := SALDO + VALOR;
            }
            senao
            {
                  se ( TIPO == 'r' | TIPO == 'R' )
                  {
                        SALDO := SALDO - VALOR;
                  }
            }

            // verifica situacao do balanco minimo

            se ( SALDO < MINIMO )
            {
                  SALDO := SALDO - SALDO * TAXA / 100;
                  escreva ( "Conta ", CONTA , " => NAO HA FUNDOS" );
            }
            senao
            {
                  escreva ( "Conta ", CONTA, " => R$ ", SALDO );
            }
            TRANS := TRANS - 1;
      }
}
```

Problema 1.12.16

Uma empresa decidiu fazer um levantamento em relação aos candidatos que se apresentarem para preenchimento de vagas no seu quadro de funcionários, utilizando processamento eletrônico. Supondo que você seja o programador encarregado desse levantamento, fazer um algoritmo que:

- leia um conjunto de dados para cada candidato contendo:
  a) numero de inscrição do candidato
  b) idade
  c) sexo (masculino, feminino)
  d) experiência no serviço (sim, não)

O último conjunto contem o numero de inscrição do candidato igual a zero.

- calcule
  a)o número de candidatos do sexo feminino
  b)o número de candidatos do sexo masculino
  c)idade média dos homens que já tem experiência no serviço
  d)porcentagem dos homens com mais de 45 anos entre o total de homens
  e)número de mulheres que tem idade inferior a 35 anos e com experiência no serviço
  f)a menor idade entre as mulheres que já tem experiência no serviço
- escreva
  a) o número de inscrição das mulheres pertencentes ao grupo descrito no item e)
  b) o que foi calculado em cada item acima especificado

```
algoritmo()
{
        declare INSCRICAO:inteiro;      // numero de inscricao do candidato
        declare IDADE:inteiro;          // idade do candidato
        declare SEXO:caracter;          // sexo do candidato (m,f,M,F)
        declare EXPERIENCIA:caracter;   // ja tem experiencia (s,n,S,N)
        declare QTD_FEM:inteiro;        // numero de candidatos do sexo feminino
        declare QTD_MAS:inteiro;        // numero de candidatos do sexo masculino
        declare ID_MEDIA:inteiro;
                // idade media dos homens que ja tem experiencia
        declare S_IDADE:inteiro;
                // soma das idades dos homens com experiencia
        declare HOM_EXP:inteiro;        // numero de homens com experiencia
        declare PORCENT:real;
                // porcentagem de homens com mais de 45 anos
        declare HOM_45:inteiro;   // numero de homens com mais de 45 anos
        declare ID_MENOR:inteiro;
                // menor idade entre as mulheres com experiencia
        declare MUL_EXP:inteiro;
                // numero de mulheres menor de 35 e com exp

        // inicializacao dos acumuladores

        QTD_FEM := 0;
        QTD_MAS := 0;
```

```
S_IDADE := 0;
HOM_EXP := 0;
HOM_45 := 0;

// incializacao das condicoes limites

ID_MENOR := 100;          // a menor idade

// incializacao da condicao de contorno

INSCRICAO := 1;

enquanto ( INSCRICAO <> 0 )
{
        // recebe o numero de inscricao

        leia ( "informe o numero de inscricao: ", INSCRICAO );

        se ( INSCRICAO <> 0 )
        {
                // entrada das informacoes complementares

                leia ( "informe a idade: ", IDADE );
                leia ( "informe o sexo: ", SEXO );
                leia ( "ja tem experiencia (s/n): ", EXPERIENCIA );

                // calculo do numero de candidatos em funcao do sexo

                se ( SEXO == 'M' | SEXO == 'm' )
                {
                        QTD_MAS := QTD_MAS + 1;

                        // verifica se ja tem experiencia

                        se ( EXPERIENCIA == 'S' | EXPERIENCIA == 's' )
                        {
                                HOM_EXP := HOM_EXP + 1;
                                S_IDADE := S_IDADE + IDADE;
                        }

                        // verifica se ja tem mais de 45 anos

                        se ( IDADE > 45 )
                        {
                                HOM_45 := HOM_45 + 1;
```

```
                        }
                }
                senao
                {
                        se ( SEXO == 'F' | SEXO == 'f' )
                        {
                                QTD_FEM := QTD_FEM + 1;

                                // mulheres com experiencia

                                se ( EXPERIENCIA == 'S' | EXPERIENCIA == 's' )
                                {
                                        // a menor idade

                                        se ( ID_MENOR < IDADE )
                                        {
                                                ID_MENOR := IDADE;
                                        }

                                        // mulheres com menos de 35 anos

                                        se ( IDADE < 35 )
                                        {
                                                MUL_EXP := MUL_EXP + 1;
                                                escreva ( "mulher experiente: ",
INSCRICAO );
                                        }
                                }
                        }
                }
        }
        }

        // calculo da idade media dos homens

        se ( HOM_EXP <> 0 )
        {
                ID_MEDIA := S_IDADE / HOM_EXP;
        }
        senao
        {
                ID_MEDIA := 0;
        }

        // calculo da porcentagem dos homens
```

```
        se ( QTD_MAS <> 0 )
        {
                PORCENT := HOM_45 / QTD_MAS;
        }
        senao
        {
                PORCENT := 0;
        }

        // saida dos resultados

        escreva ( "numero de candidatos do sexo feminino: ", QTD_FEM );
        escreva ( "numero de candidatos do sexo masculino: ", QTD_MAS );
        escreva ( "idade media dos homens > 45 e com experiencia: ", ID_MEDIA );
        escreva ( "numero de mulheres com < 35 e com experiencia: ", MUL_EXP );
        escreva ( "menor idade entre mulheres com experiencia: ", ID_MENOR );
}
```

Problema 1.12.17

Uma companhia de teatro planeja dar uma série de espetáculos. A direção calcula que a $5,00 o ingresso, serão vendidos 120 ingressos, e as despesas montarão em $200,00. A uma diminuição de $0,50 no preço dos ingressos espera-se que haja um aumento de 26 ingressos vendidos.
Fazer um algoritmo que escreva uma tabela de valores do lucro esperado em função do preço do ingresso, fazendo-se variar este preço de $5,00 a $1,00 de $0,50 em $0,50. Escreva, ainda, o lucro máximo esperado, o preço e o número de ingressos correspondentes.

```
algoritmo()
{
        declare LUCRO:real;                 // lucro esperado
        declare PRECO:real;                 // preco do ingressoo
        declare INGRESSOS_MAX:real;  // lucro maximo esperado
        declare INGRESSOS:inteiro;          // numero de ingressos correspondentes
        declare DESPESA:real;               // valor das despesas
        declare PRECO_MAX:real;
        declare LUCRO_MAX:real;

        // inicializacao dos valores iniciais

        PRECO := 5.0;
        INGRESSOS := 120;
        DESPESA := 200;

        // inicializacao das condicoes limites

        LUCRO_MAX := 0;

        // resolucao supondo que as despesas permanecam constantes

        repita ateque ( PRECO < 0.50 )
        {
                LUCRO := ( INGRESSOS * PRECO ) - DESPESA;
                escreva ( "Preco: ", PRECO,
                        " Ingressos: ", INGRESSOS, " Lucro: ", LUCRO );
                se ( LUCRO > LUCRO_MAX )
                {
                        INGRESSOS_MAX := INGRESSOS;
                        PRECO_MAX := PRECO;
                        LUCRO_MAX := LUCRO;
                }
                PRECO := PRECO - 0.5;
                INGRESSOS := INGRESSOS + 26;
```

```
        }
        escreva ( "Preco: ", PRECO_MAX,
                        " Ingressos: ", INGRESSOS_MAX, " Lucro: ", LUCRO_MAX );
}
```

Problema 1.12.18

A comissão organizadora de um rallye automobilístico decidiu apurar os resultados da competição através de um processamento eletrônico. Um dos algoritmos necessários para a classificação das equipes concorrentes e o que emite uma listagem geral do desempenho das equipes, atribuindo pontos segundo determinadas normas:
O algoritmo deverá:

- a) Ler
  - a.1) uma linha contendo os tempos padrão (em minutos decimais) para as três fases de competição
  - a.2) um conjunto de linhas contendo cada uma o numero de inscrição da equipe e os tempos (em minutos decimais) que as mesmas despenderam ao cumprir as três diferentes etapas. A ultima linha (flag), que não entrará nos cálculos, contem o número 9999 como número de inscrição.
- b) Calcular:
  - b.1) os pontos de cada equipe em cada uma das etapas, seguindo o seguinte critério:
    Seja d o valor absoluto da diferença entre o tempo padrão (lido na primeira linha) e o tempo despendido pela equipe numa etapa:
    - d < 3 minutos => atribuir 100 pontos a etapa
    - 3 <= d <= 5 minutos => atribuir 80 pontos a etapa
    - d > 5 minutos => atribuir 80 - (d-5)/5 pontos a etapa
  - b.2) o total de pontos de cada equipe nas três etapas
  - b.3) a equipe vencedora
- c) Escrever:
  - c.1) para cada equipe, o número de inscrição, os pontos obtidos em cada etapa e o total de pontos obtidos.

```
algoritmo()
{
        declare PADRAO1: real;          // tempo padrao da etapa 1
        declare PADRAO2: real;          // tempo padrao da etapa 2
        declare PADRAO3: real;          // tempo padrao da etapa 3
        declare PADRAO: real;           // tempo padrao generico
        declare INSCRICAO: inteiro;     // numero de inscricao da equipe
        declare TEMPO: real;            // tempo da equipe ao cumprir a etapa 1
        declare PONTOS: real;           // total de pontos da equipe na etapa 1
        declare TOTAL: real;            // total de pontos de cada equipe ao final
        declare CAMPEAO: inteiro;       // numero de inscricao da equipe vencedora
        declare D: real;            // diferencao entre o tempo gasto e o padrao
        declare i: inteiro;             // auxiliar
        declare MAIOR: inteiro;         // maior soma de pontos

        // inicializacao das condicoes limites

        MAIOR := 0;

        // leitura dos tempos padrao
```

```
leia ( "informe o tempo padrao para a etapa 1: ", PADRAO1 );
leia ( "informe o tempo padrao para a etapa 2: ", PADRAO2 );
leia ( "informe o tempo padrao para a etapa 3: ", PADRAO3 );

// inicializacao da condicao de contorno

INSCRICAO := 1;

repita ateque ( INSCRICAO == 9999 )
{
        // leitura do numero de inscricao

        leia ( "informe o numero de inscricao: ", INSCRICAO );

        // inicializa o total de pontos de cada equipe

        PONTOS := 0;
        TOTAL := 0;

        se ( INSCRICAO <> 9999 )
        {
                // inicio de cada etapa

                i := 1;

                repita ateque ( i > 3 )
                {
                        escreva ( "Etapa numero: ", i );
                        leia ( "informe o tempo gasto na etapa: ", TEMPO );

                        // calculo dos pontos obtidos pela equipe

                        selecao
                        {
                                caso ( i == 1 )
                                {
                                        PADRAO := PADRAO1;
                                }
                                caso ( i == 2 )
                                {
                                        PADRAO := PADRAO2;
                                }
                                caso ( i == 3 )
                                {
                                        PADRAO := PADRAO3;
```

```
                                        }
                                }
                                D := TEMPO - PADRAO;
                                D := ValorAbs(D);
                                se ( D < 3 )
                                {
                                        PONTOS := 100;
                                }
                                senao
                                {
                                        se ( D >= 3 & D <= 5 )
                                        {
                                                PONTOS := 80;
                                        }
                                        senao
                                        {
                                                PONTOS := 80 - ( ( D - 5 ) / 5 );
                                        }
                                }
                                escreva ( "A equipe ", INSCRICAO, " obteve ", PONTOS,
" na etapa ", i );

                                // totaliza os pontos da equipe

                                TOTAL := TOTAL + PONTOS;

                                i := i + 1;
                        }
                        escreva ( "Total de pontos da equipe ", INSCRICAO, " igual a ",
TOTAL );
                        se ( TOTAL > MAIOR )
                        {
                                CAMPEAO := INSCRICAO;
                        }
                }
        }
        escreva ( "A equipe campea foi a equipe: ", CAMPEAO );
}
```

Problema 1.12.19

Numa certa loja de eletrodomésticos, o comerciante encarregado da seção de televisores recebe, mensalmente, um salário fixo mais comissão. Essa comissão é calculada em relação ao tipo e ao número de televisores vendidos por mês, obedecendo ao seguinte critério:

Tipo Televisor	Número de televisores Vendidos	Comissão / Televisor
a cores	maior ou igual a 10	$100,00
	menor que 10	$ 50,00
preto e branco	maior ou igual a 20	$ 40,00
	menor que 20	$ 20,00

Sabe-se, ainda, que ele tem um desconto de 8% sobre seu salário fixo para o INPS. Se o salário total (fixo + comissões - INPS) for maior ou igual a $3.000,00 ele ainda terá um desconto de 5%, sobre esse salário total, relativo ao imposto de renda retido na fonte. Sabendo-se que existem 20 empregados nesta seção, leia o valor do salário fixo e, para cada comerciante, o número de sua inscrição, o número de televisores a cores e o numero de televisores preto e branco vendidos, calcule e escreva o número de inscrição de cada empregado, seu salário bruto e seu salário líquido

```
algoritmo()
{
        declare FIXO:real;          // fixo recebido mensalmente pelo vendedor
        declare COMISSAO:real;          // comissao recebida pelo vendedor
        declare CORES:inteiro;          // televisores a cores vendidos / vendedor
        declare PRETO:inteiro;      // numero de televisores preto e branco vendidos
        declare TIPO:caracter;      // tipo de televisor (c-cores,p-preto e branco)
        declare INPS:real;              // desconto de 8% sobre o salario total
        declare BRUTO:real;             // salario total
        declare RETIDO:real;            // 5% sobre o TOTAL de retencao da fonte
        declare N:inteiro;              // numero de empregados na secao
        declare INSCRICAO:inteiro;      // numero de inscricao
        declare LIQUIDO:real;           // salario liquido

        // inicializacao dos acumuladores

        N := 0;

        // recebendo o salario fixo geral

        leia ( "informe o valor do salario fixo: ", FIXO );

        // calculo do INPS

        INPS := FIXO * 8 / 100;
```

```
// calculos para cada um dos 20 empregados

repita ateque ( N == 5 )
{
        // inicializacao dos valores para cada funcionario

        BRUTO := 0;
        LIQUIDO := 0;
        COMISSAO := 0;

        leia ( "informe o numero de inscricao: ", INSCRICAO );
        leia ( "numero de televisores a cores vendidos: ", CORES );
        leia ( "numero de televisores preto e branco vendidos: ", PRETO );

        // calculo da comissao

        se ( CORES >= 10 )
        {
                COMISSAO := 100 * CORES;
        }
        senao
        {
                COMISSAO := 50 * CORES;
        }
        se ( PRETO >= 20 )
        {
                COMISSAO := COMISSAO + 40 * PRETO;
        }
        senao
        {
                COMISSAO := COMISSAO + 20 * PRETO;
        }

        // calculo do salario bruto

        BRUTO := FIXO + COMISSAO - INPS;

        // verificacao da retencao na fonte

        se ( BRUTO > 3000 )
        {
                LIQUIDO := BRUTO - ( BRUTO * 5 / 100 );
        }
        senao
```

```
            {
                LIQUIDO := BRUTO;
            }

            // resultados

            escreva ( "Inscricao: ", INSCRICAO,
                " Bruto: ", BRUTO, " Liquido: ", LIQUIDO );

            // conta o funcionario

            N := N + 1;
        }
}
```

Problema 1.12.20

O dia da semana para uma data qualquer pode ser calculado pela seguinte fórmula:
dia = Resto(Quociente((2.6 * M - 0.2),1) + D + A + Quociente(A,4)+ Quociente(S,4) – 2 * S),7)

onde,

M, representa o número do mês. (janeiro e fevereiro são os meses 11 e 12 do ano precedente, marco e o mês 1 e dezembro e o mês 10
D, representa o dia do mês
A, representa o número formado pelos dois últimos algarismos do ano
S, representa o número formado pelos dois primeiros algarismos do ano

Os dias da semana são numerados de 0 a 6, domingo corresponde a 0, segunda a 1 e assim por diante.
Fazer um algoritmo que:

- leia um conjunto de 50 datas (dia, mês e ano)
- determine o dia da semana correspondente a data lida, segundo o método especificado
- escreva, para cada data lida, o dia, o mês, o ano e o dia da semana

Problema 1.12.21

Numa fábrica trabalham homens e mulheres divididos em três classes:

- A - os que fazem ate 30 pecas por mês
- B - os que fazem de 31 a 35 pecas por mês
- C - os fazem mais de 35 pecas por mês

A classe A recebe salário mínimo. A classe B recebe salário mínimo e mais 3% do salário mínimo por peca acima das 30 iniciais. A classe C recebe salário mínimo e mais 5% do salário mínimo por peca acima das 30 iniciais.
Fazer um algoritmo que:

- a) leia varias linhas, contendo cada uma:
  - o número do operário
  - o número de pecas fabricados por mês
  - o sexo do operário
- b) calcule e escreva:
  - o salário de cada operário
  - o total da folha mensal de pagamento da fabrica
  - o numero total de pecas fabricadas por mês
  - a media de pecas fabricadas pelos homens em cada classe
  - a media de pecas fabricadas pelas mulheres em cada classe
  - o número do operário ou operária de maior salário (não existe empate)

observação: A última linha, que servirá de flag, terá o número do operário igual a zero.

```
algoritmo()
{
        declare OPERARIO:inteiro;    // numero do operario
        declare PRODUCAO:inteiro;    // numero de pecas fabricadas por operario
        declare SEXO:caracter;       // sexo do operario (m,f)
        declare SALARIO:real;        // salario de cada operario
        declare FOLHA:real;          // total mensal da folha de pagamento
        declare PECAS_MES:inteiro;   // total de pecas fabricadas por mês
        declare MEDIA_HA:real;       // media de pecas fabricadas pelos homens da classe A
        declare MEDIA_HB:real;       // media de pecas fabricadas pelos homens da classe B
        declare MEDIA_HC:real;       // media de pecas fabricadas pelos homens da classe C
        declare MEDIA_MA:real;       // media de pecas fabricadas por mulheres da classe A
        declare MEDIA_MB:real;       // media de pecas fabricadas por mulheres da classe B
        declare MEDIA_MC:real;       // media de pecas fabricadas pelos mulheres da classe C
        declare HOMENS_A:inteiro;    // numero de homens da classe A
        declare HOMENS_B:inteiro;    // numero de homens da classe B
        declare HOMENS_C:inteiro;    // numero de homens da classe C
        declare MULHERES_A:inteiro;          // numero de mulheres da classe A
        declare MULHERES_B:inteiro;          // numero de mulheres da classe B
        declare MULHERES_C:inteiro;          // numero de mulheres da classe C
        declare PECAS_HA:inteiro;    // numero de pecas fabricadas por homens da classe A
        declare PECAS_HB:inteiro;    // numero de pecas fabricadas pelos homens da classe B
        declare PECAS_HC:inteiro;    // numero de pecas fabricadas pelos homens da classe C
        declare PECAS_MA:inteiro;    // qtd de pecas fabricadas pelas mulheres da classe A
        declare PECAS_MB:inteiro;    // qtd de pecas fabricadas pelas mulheres da classe B
        declare PECAS_MC:inteiro;    // qtd de pecas fabricadas pelas mulheres da classe C
        declare HOMENS:inteiro;              // numero total de homens
```

```
declare MULHERES:inteiro;          // numero total de mulheres
declare ID_MAIOR:inteiro;          // numero do operario(a) de maior salario
declare MAIOR:inteiro;             // maior salario
declare MINIMO:real;               // valor do salario minimo

// inicializacao dos acumuladores globais

FOLHA := 0;
PECAS_MES := 0;
HOMENS := 0;
MULHERES := 0;
HOMENS_A := 0;
HOMENS_B := 0;
HOMENS_C := 0;
MULHERES_A := 0;
MULHERES_B := 0;
MULHERES_C := 0;
PECAS_HA := 0;
PECAS_HB := 0;
PECAS_HC := 0;
PECAS_MA := 0;
PECAS_MB := 0;
PECAS_MC := 0;

// inicializacao das condicoes de extremos

MAIOR := 0;

// recebe valor do salario minimo

leia ( "informe o valor do salario minimo: ", MINIMO );

// inicializacao da condicao de contorno

OPERARIO := 1;

repita ateque ( OPERARIO == 0 )
{
        // inicializacao dos acumuladores locais

        SALARIO := 0;

        leia ( "informe o numero do operario: ", OPERARIO );
        se ( OPERARIO <> 0 )
        {
                leia ( "informe o numero de pecas fabricadas: ", PRODUCAO );
                leia ( "informe o sexo do operario: ", SEXO );

                // calculo do salario do operario
```

```
se ( PRODUCAO <= 30 )
{
        SALARIO := MINIMO;
}
senao
{
        se ( PRODUCAO >= 31 & PRODUCAO <= 35 )
        {
                SALARIO := MINIMO + ( PRODUCAO - 30 ) * 3 / 100;
        }
        senao
        {
                SALARIO := MINIMO + ( PRODUCAO - 30 ) * 5 / 100;
        }
}

// acumula o total da folha

FOLHA := FOLHA + SALARIO;

// acumula o total de pecas fabricadas por mês

PECAS_MES := PECAS_MES + PRODUCAO;

// identifica operario de maior salario

se ( SALARIO > MAIOR )
{
        MAIOR := SALARIO;
        ID_MAIOR := OPERARIO;
}

// totaliza valores para o calculo das medias

se ( SEXO == 'M' | SEXO == 'm' )
{
        selecao
        {
                caso ( PRODUCAO <= 30 )
                {
                        PECAS_HA := PECAS_HA + PRODUCAO;
                        HOMENS_A := HOMENS_A + 1;
                }
                caso ( PRODUCAO >= 31 & PRODUCAO <= 35 )
                {
                        PECAS_HB := PECAS_HB + PRODUCAO;
                        HOMENS_B := HOMENS_B + 1;
                }
                caso ( PRODUCAO > 35 )
                {
```

```
                                        PECAS_HC := PECAS_HC + PRODUCAO;
                                        HOMENS_C := HOMENS_C + 1;
                                }
                        }
                }
                senao
                {
                        selecao
                        {
                                caso ( PRODUCAO <= 30 )
                                {
                                        PECAS_MA := PECAS_MA + PRODUCAO;
                                        MULHERES_A := MULHERES_A + 1;
                                }
                                caso ( PRODUCAO >= 31 & PRODUCAO <= 35 )
                                {
                                        PECAS_MB := PECAS_MB + PRODUCAO;
                                        MULHERES_B := MULHERES_B + 1;
                                }
                                caso ( PRODUCAO > 35 )
                                {
                                        PECAS_MC := PECAS_MC + PRODUCAO;
                                        MULHERES_C := MULHERES_C + 1;
                                }
                        }
                }

                // salario do operario

                escreva ( "O operario ", OPERARIO, " recebera ", SALARIO );

        }
}

// calculo das medias

MEDIA_HA := PECAS_HA / HOMENS_A;
MEDIA_HB := PECAS_HB / HOMENS_B;
MEDIA_HC := PECAS_HC / HOMENS_C;
MEDIA_MA := PECAS_MA / MULHERES_A;
MEDIA_MB := PECAS_MB / MULHERES_B;
MEDIA_MC := PECAS_MC / MULHERES_C;

// resultados

escreva ( "Total da folha de pagamento: ", FOLHA );
escreva ( "Producao do mes: ", PECAS_MES );
escreva ( "Media de producao dos homens da classe A: ", MEDIA_HA );
escreva ( "Media de producao dos homens da classe B: ", MEDIA_HB );
escreva ( "Media de producao dos homens da classe C: ", MEDIA_HC );
```

```
        escreva ( "Media de producao das mulheres da classe A: ", MEDIA_MA );
        escreva ( "Media de producao das mulheres da classe B: ", MEDIA_MB );
        escreva ( "Media de producao das mulheres da classe C: ", MEDIA_MC );
        escreva ( "Operario de maior salario: ", ID_MAIOR );
}
```

Problema 1.12.22

Uma determinada fábrica de rádios possui duas linhas de montagem distintas standard e luxo. A linha de montagem standard comporta um máximo de 24 operários, cada rádio standard da um lucro X e gasta um homem-dia para sua confecção. A linha de montagem luxo comporta no máximo 32 operários e cada radio da um lucro Y e gasta 2 homens-dia para a sua confecção. A fábrica possui 40 operários. O mercado e capaz de absorver toda a produção e o fabricante deseja saber qual esquema de produção a adotar de modo a maximizar seu lucro diário.
Fazer um algoritmo que leia os valores de X e Y e escreva, para esse esquema de lucro máximo, o numero de operários na linha standard e na linha luxo, o numero de rádios standard e luxo produzidos e o lucro.

```
algoritmo()
{
        declare X:real;                 // lucro do radio standard (unitario)
        declare Y:real;                 // lucro do radio luxo (unitario)
        declare STANDARD:inteiro;       // numero de radios standard produzidos
        declare LUXO:inteiro;           // numero de radios luxo produzidos
        declare LUCRO:real;             // valor do lucro auferido
        declare OPERARIOS:inteiro;      // numero de operarios da fabrica
        declare LINHA_STAN:inteiro;     // numero de operarios na linha standard
        declare LINHA_LUXO:inteiro;     // numero de operarios na linha luxo
        declare HOMENS_STAN:inteiro;    // homens-dia para a confeccao do standard
        declare HOMENS_LUXO:inteiro;    // homens-dia para a confeccao do luxo
        declare LUCRO_MAX:real;         // maior lucro

        // inicializacao dos parametros iniciais

        OPERARIOS := 40;
        LUCRO_MAX := 0;

        // recebendo os valores de X e Y

        leia ( "informe o lucro do radio standard: ", X );
        leia ( "informe o lucro do radio luxo: ", Y );

        /* consideracoes sobre o problema

        1.expressao do lucro total

                LUCRO := STANDARD * X + LUXO * Y;

        2.limites da linha de producao

                HOMENS_STAN <= 24, HOMENS_LUXO <= 32,
                HOMENS_STAN + HOMENS_LUXO <= 40 (numero de operarios da fabrica)
                HOMENS_STAN_MIN = 40 - HOMENS_LUXO = 40 - 32 = 8
                HOMENS_LUXO_MIN = 40 - HOMENS_STAN = 40 - 24 = 16
```

```
        3.producao em homens-dia

                STANDARD = HOMENS_STAN, LUXO = HOMENS_LUXO / 2

        4.colocando a expressao do lucro em funcao do numero de homens

                LUCRO := HOMENS_STAN * X + HOMENS_LUXO * Y / 2 e
                HOMENS_LUXO := 40 - HOMENS_STAN

*****/

        // inicializacao do condicao de contorno do laco

        HOMENS_STAN := 8;

        repita ateque ( HOMENS_STAN == 24 )
        {
                HOMENS_LUXO := OPERARIOS - HOMENS_STAN;
                LUCRO := HOMENS_STAN * X + ( HOMENS_LUXO * Y ) / 2;
                se ( LUCRO > LUCRO_MAX )
                {
                        LUCRO_MAX := LUCRO;
                        LINHA_STAN := HOMENS_STAN;
                        LINHA_LUXO := HOMENS_LUXO;
                }
                HOMENS_STAN := HOMENS_STAN + 1;
        }

        HOMENS_LUXO := 16;

        repita ateque ( HOMENS_LUXO == 32 )
        {
                HOMENS_STAN := OPERARIOS - HOMENS_LUXO;
                LUCRO := HOMENS_STAN * X + ( HOMENS_LUXO * Y ) / 2;
                se ( LUCRO > LUCRO_MAX )
                {
                        LUCRO_MAX := LUCRO;
                        LINHA_STAN := HOMENS_STAN;
                        LINHA_LUXO := HOMENS_LUXO;
                }
                HOMENS_LUXO := HOMENS_LUXO + 1;
        }

        STANDARD := HOMENS_STAN;
        LUXO := HOMENS_LUXO / 2;
        escreva ( "Numero de operarios na linha standard: ", LINHA_STAN );
        escreva ( "Numero de operarios na linha luxo: ", LINHA_LUXO );
        escreva ( "Numero de radios standard produzidos: ", STANDARD );
        escreva ( "Numero de radios luxo produzidos: ", LUXO );
        escreva ( "Lucro obtido: ", LUCRO_MAX );
```

```
}
```

Problema 1.12.23

Fazer um algoritmo para calcular o numero de dias decorridos entre duas datas (considerar também a ocorrência de anos bissextos), sabendo-se que:
a) cada par de datas é lido numa linha, a última linha contem o número do dia negativo
b) a primeira data na linha é sempre a mais antiga e o ano está digitado com 4 dígitos.

```
algoritmo()
{
        declare DIA1: inteiro;
        declare MES1: inteiro;
        declare ANO1: inteiro;
        declare DIA2: inteiro;
        declare MES2: inteiro;
        declare ANO2: inteiro;
        declare DIFE: inteiro;                  // numero de dias entre as duas datas
        declare BISSEXTO:inteiro;
        declare AUX: inteiro;

        // recebendo as datas

        leia ( "(data 1)  informe o dia: ", dia1 );
        leia ( "          informe o mes : ", mes1 );
        leia ( "          informe o ano : ", ano1 );
        leia ( "(data 2) informe o dia : ", dia2 );
        leia ( "          informe o mes : ", mes2 );
        leia ( "          informe o ano : ", ano2 );

        // inicializando os acumuladores

        DIFE := 0;

        // acumulando os anos

        AUX := ANO1;

        repita ateque ( AUX == ANO2 )
        {
                BISSEXTO := Resto ( AUX, 4 );
                se ( BISSEXTO == 0 )
                {
                        DIFE := DIFE + 366;
                }
                senao
                {
                        DIFE := DIFE + 365;
                }
```

```
                AUX := AUX + 1;
        }
        escreva ( "A diferenca em dias entre ", DIA1, "/", MES1, "/", ANO1, " e ", DIA2, "/",
                        MES2, "/", ANO2, " , ", DIFE );

        /* ainda esta incompleto ... */

}
```

Problemas envolvendo o cálculo de somatórios

Problema 1.12.24

Fazer um algoritmo que calcule e escreva o valor de S:

$$S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + ... + \frac{99}{50}$$

```
algoritmo()
{
        declare S:real;             // o valor do somatorio
        declare NUM:inteiro;        // numerador da expressao
        declare DEN:inteiro;        // denominador da expressao
        declare T:real;             // cada um dos termos do somatorio

        // inicializacao dos acumuladores

        S := 0;

        // inicializacao das condicoes iniciais

        NUM := 1;
        DEN := 1;

        // faz o calculo do somatorio

        repita ateque ( NUM > 99 )
        {
                T := NUM / DEN;
                S := S + T;
                NUM := NUM + 2;
                DEN := DEN + 1;
        }

        // resultado

        escreva ( "O valor de S e ", S );
}
```

Problema 1.12.24a

Fazer um algoritmo que calcule e escreva o valor de S:

$$S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + ... + \frac{99}{50}$$

```
algoritmo( )
{
        declare S:real;             // o valor do somatorio
        declare NUM:inteiro;        // numerador da expressao
        declare DEN:inteiro;        // denominador da expressao

        // inicializacao dos acumuladores

        S := 0;

        // inicializacao das condicoes iniciais

        NUM := 1;
        DEN := 1;

        // faz o calculo do somatorio

        repita ateque ( NUM > 99 )
        {
                S := S + NUM / DEN;
                NUM := NUM + 2;
                DEN := DEN + 1;
        }

        // resultado

        escreva ( "O valor de S e ", S );
}
```

Problema 1.12.24b

Fazer um algoritmo que calcule e escreva o valor de S:

$$S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + ... + \frac{99}{50}$$

```
algoritmo( )
{
        declare S:real;         // o valor do somatorio
        declare N:inteiro;      // numerador da expressao

        // inicializacao dos acumuladores

        S := 0;

        // inicializacao das condicoes iniciais

        N := 1;

        // faz o calculo do somatorio

        repita ateque ( N > 50 )
        {
                S := S + ( ( 2*N - 1 ) / N );
                N := N + 1;
        }

        // resultado

        escreva ( "O valor de S e ", S );
}
```

Problema 1.12.25

Fazer um algoritmo que calcule e escreva o valor de S:

$$S = \frac{2^1}{50} + \frac{2^2}{49} + \frac{2^3}{48} + ... + \frac{2^{50}}{1}$$

```
algoritmo( )
{
        declare S:real;                 // o valor do somatorio
        declare DEN:inteiro;            // denominador da expressao
        declare T:real;                 // cada um dos termos do somatorio
        declare i:inteiro;              // expoente

        // inicializacao dos acumuladores

        S := 0;

        // inicializacao das condicoes iniciais

        DEN := 50;
        i := 1;

        // faz o calculo do somatorio

        repita ateque ( i > 50 )
        {
                T := ( 2 ^ i ) / DEN;
                S := S + T;
                i := i + 1;
                DEN := DEN - 1;
        }

        // resultado

        escreva ( "O valor de S e ", S );
}
```

Problema 1.12.25a

Fazer um algoritmo que calcule e escreva o valor de S:

$$S = \frac{2^1}{50} + \frac{2^2}{49} + \frac{2^3}{48} + \ldots + \frac{2^{50}}{1}$$

```
algoritmo()
{
        declare S:real;                 // o valor do somatorio
        declare DEN:inteiro;            // denominador da expressao
        declare T:real;                 // cada um dos termos do somatorio

        // inicializacao dos acumuladores

        S := 0;

        // inicializacao das condicoes iniciais

        DEN := 50;

        // faz o calculo do somatorio

        repita ateque ( DEN == 0 )
        {
                T := ( 2 ^ ( 51 - DEN ) ) / DEN;
                S := S + T;
                DEN := DEN - 1;
        }

        // resultado

        escreva ( "O valor de S e ", S );
}
```

Problema 1.12.26

Fazer um algoritmo que calcule e escreva o valor de S:

$$S = \frac{37 \times 38}{1} + \frac{36 \times 37}{2} + \frac{35 \times 36}{3} + ... + \frac{1 \times 2}{37}$$

```
algoritmo()
{
        declare S:real;             // o valor do somatorio
        declare NUM:real;           // numerador da expressao
        declare DEN:inteiro;        // denominador da expressao
        declare T:real;             // cada um dos termos do somatorio

        // inicializacao dos acumuladores

        S := 0;

        // inicializacao das condicoes iniciais

        NUM := 37;
        DEN := 1;

        // faz o calculo do somatorio

        repita ateque ( NUM < 1 )
        {
                T := ( NUM * ( NUM + 1 ) ) / DEN;
                S := S + T;
                NUM := NUM - 1;
                DEN := DEN + 1;
        }

        // resultado

        escreva ( "O valor de S e ", S );
}
```

Problema 1.12.27

Fazer um algoritmo que calcule e escreva o valor de S:

$$S = \frac{1}{1} - \frac{2}{4} + \frac{3}{9} - \frac{4}{16} + \frac{5}{25} - \frac{6}{36} + \ldots - \frac{10}{100}$$

```
algoritmo()
{
        declare S:real;             // o valor do somatorio
        declare NUM:real;           // numerador da expressao
        declare DEN:inteiro;        // denominador da expressao
        declare T:real;             // cada um dos termos do somatorio
        declare AUX:inteiro;        // auxiliar para decidir sobre o sinal de T

        // inicializacao dos acumuladores

        S := 0;

        // inicializacao das condicoes iniciais

        NUM := 1;
        DEN := 1;

        // faz o calculo do somatorio

        enquanto ( NUM <= 10 )
        {
                T := NUM / DEN;
                AUX := Resto ( DEN, 2 );
                se ( AUX <> 0 )
                {
                        S := S + T;
                }
                senao
                {
                        S := S - T;
                }
                NUM := NUM + 1;
                DEN := NUM ^ 2;
        }

        // resultado

        escreva ( "O valor de S e ", S );
```

```
}
```

Problema 1.12.27a

Fazer um algoritmo que calcule e escreva o valor de S:

$$S = \frac{1}{1} - \frac{2}{4} + \frac{3}{9} - \frac{4}{16} + \frac{5}{25} - \frac{6}{36} + \ldots - \frac{10}{100}$$

```
algoritmo()
{
        declare S:real;         // o valor do somatorio
        declare N:real;         // numerador da expressao

        // inicializacao dos acumuladores

        S := 0;

        // inicializacao das condicoes iniciais

        N := 1;

        // faz o calculo do somatorio

        enquanto ( N <= 10 )
        {
                se ( N % 2 <> 0 )
                {
                        S := S + ( N / ( N ^ 2 ) );
                }
                senao
                {
                        S := S - N / ( N ^ 2 ) );
                }
                N := N + 1;
        }

        // resultado

        escreva ( "O valor de S e ", S );
}
```

Problema 1.12.28

Fazer um algoritmo que calcule e escreva a soma dos 50 primeiros termos da seguinte série:

$$S = \frac{1000}{1} - \frac{997}{2} + \frac{994}{3} - \frac{991}{4} + \ldots$$

```
algoritmo( )
{
        declare S:real;             // o valor do somatorio
        declare NUM:real;           // numerador da expressao
        declare DEN:inteiro;        // denominador da expressao
        declare T:real;             // cada um dos termos do somatorio
        declare AUX:inteiro;        // auxiliar para decidir sobre o sinal de T

        // inicializacao dos acumuladores

        S := 0;

        // inicializacao das condicoes iniciais

        NUM := 1000;
        DEN := 1;

        // faz o calculo do somatorio

        enquanto ( DEN <= 50 )
        {
                T := NUM / DEN;
                AUX := Resto ( DEN, 2 );
                se ( AUX <> 0 )
                {
                        S := S + T;
                }
                senao
                {
                        S := S - T;
                }
                NUM := NUM - 3;
                DEN := DEN + 1;
        }

        // resultado
```

```
        escreva ( "O valor de S e ", S );
}
```

Problema 1.12.28a

Fazer um algoritmo que calcule e escreva a soma dos 50 primeiros termos da seguinte série:

$$S = \frac{1000}{1} - \frac{997}{2} + \frac{994}{3} - \frac{991}{4} + \ldots$$

```
algoritmo()
{
        declare S:real;             // o valor do somatorio
        declare NUM:real;           // numerador da expressao
        declare DEN:inteiro;        // denominador da expressao
        declare AUX:inteiro;        // auxiliar para decidir sobre o sinal de T

        // inicializacao dos acumuladores

        S := 0;

        // inicializacao das condicoes iniciais

        NUM := 1000;
        DEN := 1;
        AUX := 1;

        // faz o calculo do somatorio

        enquanto ( DEN <= 50 )
        {
                se ( AUX == 1 )
                {
                        S := S + NUM / DEN;
                        AUX := 0;
                }
                senao
                {
                        S := S - NUM / DEN;
                        AUX := 1;
                }
                NUM := NUM - 3;
                DEN := DEN + 1;
        }

        // resultado

        escreva ( "O valor de S e ", S );
```

```
}
```

Problema 1.12.29

Fazer um algoritmo que calcule e escreva a soma dos 30 primeiros termos da seguinte série:

$$S = \frac{480}{10} - \frac{475}{11} + \frac{470}{12} - \frac{465}{13} + \ldots$$

```
algoritmo()
{
        declare S:real;             // o valor do somatorio
        declare NUM:real;           // numerador da expressao
        declare DEN:inteiro;        // denominador da expressao
        declare I:inteiro;          // contador para o numero de termos
        declare AUX:inteiro;        // auxiliar para decidir sobre o sinal de T

        // inicializacao dos acumuladores

        S := 0;
        I := 0;

        // inicializacao das condicoes iniciais

        NUM := 480;
        DEN := 10;

        // faz o calculo do somatorio

        enquanto ( I <= 30 )
        {
                AUX := Resto ( DEN, 2 );
                se ( AUX == 0 )
                {
                        S := S + NUM / DEN;
                }
                senao
                {
                        S := S - NUM / DEN;
                }
                NUM := NUM - 5;
                DEN := DEN + 1;
                I := I + 1;
        }
        // resultado

        escreva ( "O valor de S e ", S );
```

```
}
```

Problema 1.12.30

Escrever um algoritmo para gerar e escrever uma tabela com os valores do seno de um ângulo A em radianos, utilizando a série de Mac-laurim truncada apresentada a

$$\operatorname{sen} A = A - \frac{A^3}{6} + \frac{A^5}{120} - \frac{A^7}{5040}$$

seguir

condições: os valores dos ângulos A devem variar de 0.0 a 6.3, inclusive, de 0.1 em 0.1

```
algoritmo( )
{
        declare SenA:real;          // valor do seno do angulo A
        declare A:real;             // o angulo a ser gerado

        // inicializacao das condicoes iniciais

        A := 0;

        // faz o calculo do somatorio

        enquanto ( A <= 6.3 )
        {
                SenA := A - ((A^3) / 6) + ((A^5) / 120) - ((A^7) / 5040);
                escreva ( "Angulo: ", A, " Seno(A): ", SenA );
                A := A + 0.1;
        }
}
```

Problema 1.12.30a

Escrever um algoritmo para gerar e escrever uma tabela com os valores do seno de um ângulo A em radianos, utilizando a série de Mac-laurim truncada apresentada a seguir

$$\text{sen}\, A = A - \frac{A^3}{6} + \frac{A^5}{120} - \frac{A^7}{5040}$$

condições: os valores dos ângulos A devem variar de 0.0 a 6.3, inclusive, de 0.1 em 0.1

```
algoritmo()
{
        declare SenA:real;          // valor do seno do angulo A
        declare A:real;             // o angulo a ser gerado
        declare p1,p2,p3:real;      // cada uma das potencias de A

        // inicializacao das condicoes iniciais

        A := 0;

        // faz o calculo do somatorio

        enquanto ( A <= 6.3 )
        {
                p1 := Potencia (A,3);
                p2 := Potencia (A,5);
                p3 := Potencia (A,7);
                SenA := A - ( p1 / 6 ) + ( p2 / 120 ) - ( p3 / 5040 );
                escreva ( "Angulo: ", A, " Seno(A): ", SenA );
                A := A + 0.1;
        }
}
```

Problema 1.12.31

Escrever um algoritmo para gerar e escrever o valor do numero pi, com precisão de 0,0001, usando a série

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \ldots$$

Para obter a precisão desejada, adicionar apenas os termos cujo valor absoluto seja maior ou igual a 0,0001.

```
algoritmo()
{
        declare PI:real;            // valor de pi
        declare N:real;             // numerador e denominador da serie
        declare AUX:real;           // utilizado para descobrir a precisao
        declare I:inteiro;          // sinal dos termos

        // inicializacao das condicoes iniciais

        PI := 4;
        N := 3;
        I := 0;

        // faz o calculo do somatorio

        enquanto ( AUX <= 0.0001 )
        {
                AUX := 4 / N;
                se ( I == 1 )
                {
                        PI := PI + AUX;
                        I := 0;
                }
                senao
                {
                        PI := PI - AUX;
                        I := 1;
                }
                N := N + 2;
        }
        escreva ( "O valor calculado e: ", PI );
}
```

Problema 1.12.32

O valor aproximado de PI pode ser calculado usando-se a série

$$S = \frac{1}{1^3} - \frac{1}{3^3} + \frac{1}{5^3} - \frac{1}{7^3} + \frac{1}{9^3} - \ldots$$

sendo

$$\pi = \sqrt[3]{S \times 32}$$

Fazer um algoritmo para calcular e escrever o valor de PI com 51 termos

```
algoritmo()
{
        declare PI:real;            // valor de pi
        declare I:real;             // contador do numero de termos (51)
        declare AUX:real;           // utilizado para descobrir o sinal
        declare N:inteiro;          // denominador
        declare S:real;             // valor do somatorio

        // inicializacao das condicoes iniciais

        N := 1;
        I := 0;
        S := 0;
        AUX := 1;

        // faz o calculo do somatorio

        enquanto ( I <= 51 )
        {
                se ( AUX > 0 )
                {
                        S := S + ( 1 / ( N^3));
                        AUX := -1;
                }
                senao
                {
                        S := S - ( 1 / ( N^3));
                        AUX := 1;
                }
                N := N + 2;
                I := I + 1;
        }
        PI := S * 32;
        PI := Potencia(PI,1/3);
        escreva ( "O valor calculado e: ", PI );
```

```
}
```

Problema 1.12.33

Fazer um algoritmo que:
a) leia o valor de X de uma unidade de entrada

$$S = \frac{x^{25}}{1} - \frac{x^{24}}{2} + \frac{x^{23}}{3} - \frac{x^{22}}{4} + \ldots + \frac{x}{25}$$

b) calcule e escreva o valor do seguinte somatório

```
algoritmo()
{
        declare X:real;             // o numero lido
        declare AUX:real;           // utilizado para descobrir o sinal
        declare N:inteiro;          // denominador
        declare S:real;             // valor do somatorio
        declare P:real;             // potencia de X
        declare I:inteiro;          // expoente de X

        // inicializacao das condicoes iniciais

        I := 25;
        N := 1;
        S := 0;
        AUX := 1;

        leia ( "informe o valor de X: ", X );

        // faz o calculo do somatorio

        enquanto ( N <= 25 )
        {
                P := Potencia(X,I);
                P := P / N;
                se ( AUX > 0 )
                {
                        S := S + P;
                        AUX := -1;
                }
                senao
                {
                        S := S - P;
                        AUX := 1;
                }
                N := N + 1;
                I := I - 1;
```

```
        }
        escreva ( "O valor do somatorio para X = ", X, " e: ", S );
}
```

Problema 1.12.34

Fazer um algoritmo que calcule e escreva o valor de S dado por

$$S = \frac{1}{225} - \frac{2}{196} + \frac{4}{169} - \frac{8}{144} + ... + \frac{16384}{1}$$

```
algoritmo()
{
        declare S:real;                 // valor do somatorio
        declare NUM:inteiro;            // numerador da expressao
        declare DEN:inteiro;            // denomiandor da expressao
        declare AUX:real;               // utilizado para descobrir o sinal

        // inicializacao das condicoes iniciais

        S := 0;
        NUM := 1;               // PG de razao 2
        DEN := 15;              // os quadrados perfeitos menores que 15
        AUX := 1;               // quando AUX==1 os valores sao somados

        // faz o calculo do somatorio

        enquanto ( DEN >= 1 )
        {
                se ( AUX > 0 )
                {
                        S := S + NUM / (DEN^2);
                        AUX := -1;
                }
                senao
                {
                        S := S - NUM / (DEN^2);
                        AUX := 1;
                }
                NUM := NUM * 2;
                DEN := DEN - 1;
        }
        escreva ( "O valor do somatorio e: ", S );
}
```

Problema 1.12.35

Fazer um algoritmo que calcule e escreva o valor de S dado pela soma dos 20 primeiros termos da série

$$S = \frac{100}{0!} + \frac{99}{1!} + \frac{98}{2!} + \frac{97}{3!} + \ldots$$

```
algoritmo()
{
        declare S:real;             // valor do somatorio
        declare NUM:inteiro;        // numerador da expressao
        declare DEN:inteiro;        // denomiandor da expressao
        declare T:real;             // o termo generico da serie
        declare FAT:inteiro;        // o fatorial do denominador
        declare N:inteiro;          // auxiliar no calculo do fatorial
        declare I:inteiro;          // contador do numero de termos
        declare AUX:inteiro;        // auxiliar no calculo dos fatoriais

        // inicializacao dos acumuladores

        S := 0;
        I := 0;

        // inicializacao das condicoes iniciais

        NUM := 100;         // PA de razao –1
        DEN := 0;           // PA de razao 1
        AUX := 0;

        // faz o calculo do somatorio

        enquanto ( I <= 20 )
        {
                // inicializa condicoes para o calculo do fatorial

                FAT := 1;
                N := AUX;

                repita ateque ( N == 0 )
                {
                        FAT := FAT * N;
                        N := N - 1;
                }

                DEN := FAT;
```

```
                T := NUM / DEN;
                S := S + T;

                // atualiza o numerador e o denominador

                NUM := NUM - 1;

                // proximo denominador para o calculo do fatorial

                AUX := AUX + 1;

                // conta o numero de termos ja calculados

                I := I + 1;
        }
        escreva ( "O valor do somatorio e: ", S );
}
```

Problema 1.12.35a

Fazer um algoritmo que calcule e escreva o valor de S dado pela soma dos 20 primeiros termos da série

$$S = \frac{100}{0!} + \frac{99}{1!} + \frac{98}{2!} + \frac{97}{3!} + \ldots$$

```
algoritmo()
{
        declare S:real;             // valor do somatorio
        declare NUM:inteiro;        // numerador da expressao
        declare T:real;             // o termo generico da serie
        declare FAT:inteiro;        // o fatorial do denominador
        declare N:inteiro;          // auxiliar no calculo do fatorial
        declare I:inteiro;          // contador do numero de termos

        // inicializacao dos acumuladores

        S := 0;
        I := 0;

        // inicializacao das condicoes iniciais

        NUM := 100;         // PA de razao –1
        FAT := 1;           // PA de razao 1
        N := 1;

        // faz o calculo do somatorio

        enquanto ( I <= 20 )
        {
                T := NUM / FAT;             // calcula o termo generico
                S := S + T;                 // acumula o somatorio
                NUM := NUM - 1;             // atualiza o numerador
                FAT := FAT * N;             // atualiza o denominador (fatorial)
                N := N + 1;             // proximo multiplicador para denominador
                I := I + 1;             // conta o numero de termos ja calculados
        }
        escreva ( "O valor do somatorio e: ", S );
}
```

Problema 1.12.36

Elaborar um algoritmo que:
a) calcule e escreva o valor da série abaixo com precisão menor que um décimo de milionésimo (0,0000001)
b) indique quantos termos foram usados

$$S = 63 + \frac{61}{1!} + \frac{59}{2!} + \frac{57}{3!} + \ldots$$

```
algoritmo( )
{
        declare S:real;             // valor do somatorio
        declare NUM:inteiro;        // numerador da expressao
        declare FAT:inteiro;        // o fatorial do denominador
        declare T:real;             // o termo generico da serie
        declare N:inteiro;          // auxiliar no calculo do fatorial
        declare I:inteiro;          // contador do numero de termos

        // inicializacao dos acumuladores

        S := 0;
        I := 0;

        // inicializacao das condicoes iniciais

        NUM := 63;          // PA de razao –2
        FAT := 1;           // PA de razao 1
        N := 1;

        /* inicializa condicao de contorno para o laco e
                faz o calculo do somatorio */

        T := 1;

        repita ateque ( T <= 0.0000001 )
        {
                T := NUM / FAT;
                S := S + T;
                NUM := NUM - 2;
                FAT := FAT * N;
                N := N + 1;
                I := I + 1;
        }
        escreva ( "O valor do somatorio e: ", S );
        escreva ( "O numero de termos utilizados foi: ", I );
```

```
}
```

Problema 1.12.37

Fazer um algoritmo que calcule e escreva a soma dos 50 primeiros termos da série:

$$S = \frac{1!}{1} - \frac{2!}{3} + \frac{3!}{7} - \frac{4!}{15} + \frac{5!}{31} - \ldots$$

```
algoritmo()
{
        // declaracao da estrutura de dados

        declare S:real;             // valor do somatorio
        declare FAT:inteiro;        // numerador da expressao
        declare DEN:inteiro;        // o fatorial do denominador
        declare T:real;             // o termo generico da serie
        declare N:inteiro;          // auxiliar no calculo do fatorial
        declare I:inteiro;          // contador do numero de termos
        declare E:inteiro;          // expoente
        declare AUX:inteiro;        // definir o sinal do termo

        // inicializacao dos acumuladores

        S := 0;
        I := 0;

        // inicializacao das condicoes iniciais

        FAT := 1;       // PA crescente de razao 1
        DEN := 1;       // DEN + 2^E (anterior mais uma potencia de 2)
        N := 1;         // primeiro termo da serie
        E := 0;         // expoente para compor o denominador
        AUX := 1;       // os termos com numeradores impares sao somados

        // laco para o calculo do somatorio

        repita ateque ( I > 50 )
        {
                T := FAT / DEN;
                AUX := Resto ( N, 2 );      // verifica se N e par
                se ( AUX == 0 )
                {
                        T := -1 * T;        // se for, troca o sinal de T
                }
                S := S + T;             // acumula
```

```
                // proximos valores

                N := N + 1;
                E := E + 1;
                FAT := FAT * N;
                DEN := DEN + 2^E;

                // conta o numero de termos ja calculados

                I := I + 1;
        }
        escreva ( "O valor do somatorio e: ", S );
}
```

Problema 1.12.38

Fazer um algoritmo que calcule o valor de e^x através da série

$$e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

de modo que o mesmo difira do valor calculado através da função EXP de no máximo 0.0001. O valor de x deve ser lido de uma unidade de entrada. O algoritmo deverá escrever o valor de x, o valor calculado através da série, o valor dado pela função EXP e o número de termos utilizados da série

```
algoritmo()
{
        // declaracao da estrutura de dados

        declare S:real;             // valor do somatorio
        declare NUM:inteiro;        // o numerador da expressao
        declare FAT:inteiro;        // denominador da expressao
        declare T:real;             // o termo generico da serie
        declare N:inteiro;          // auxiliar no calculo do fatorial
        declare I:inteiro;          // contador do numero de termos
        declare E:inteiro;          // expoente
        declare X:inteiro;          // argumento lido
        declare K:real;             // valor de e^x obtido pela funcao EXP
        declare DIF:real;           // diferenca entre os valores

        // inicializacao dos acumuladores

        S := 0;
        I := 0;

        // inicializacao das condicoes iniciais

        FAT := 1;       // o denominador e uma PA crescente de razao 1
        NUM := 1;       // o numerador e uma potencia de x
        N := 1;         // primeiro termo da serie
        E := 0;         // expoente para compor o denominador

        // recebendo o argumento x do usuario

        leia ( "informe o valor de x: ", X );
        // calculando e^X atraves da funcao

        K := Exponencial(X);
```

```
        /*
                inicializacao da condicao de contorno para o laco e
                        laco para o calculo do somatorio
                */

        DIF := 1;

        enquanto ( DIF > 0.0001 )
        {
                T := NUM / FAT;     // calcula o termo, verificar overflow de FAT
                S := S + T;         // acumula
                DIF := K - S;       // calcula a diferenca entre os valores
                DIF := ValorAbs(DIF);       // em valor absoluto

                // gera os proximos valores da serie

                E := E + 1;         // proximo expoente do argumento X
                NUM := X^E;         // proximo numerador
                N := N + 1;         // proximo fatorial
                FAT := FAT * N;     // proximo denominador

                // conta o numero de termos ja calculados
                I := I + 1;
        }
        escreva ( "O valor do argumento x e: ", X );
        escreva ( "Valor calculado atraves da serie: ", S );
        escreva ( "Valor dado pela funcao EXP: ", K );
        escreva ( "Numero de termos utilizados: ", I );
}
```

Problema 1.12.39

Fazer um algoritmo que calcule o valor da série

$$S = x - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \ldots$$

usando os 20 primeiros termos do somatório. O valor de x é lido de uma unidade de entrada.

```
algoritmo()
{
        // declaracao da estrutura de dados

        declare X:inteiro;          // argumento lido
        declare S:real;             // valor do somatorio
        declare NUM:real;           // o numerador da expressao
        declare DEN:real;           // denominador da expressao
        declare T:real;             // o termo generico da serie
        declare N:inteiro;          // auxiliar no calculo do fatorial
        declare I:inteiro;          // contador do numero de termos
        declare E:inteiro;          // expoente
        declare AUX:inteiro;        // auxiliar no calculo do fatorial
        declare J:inteiro;          // para definir o sinal

        // inicializacao dos acumuladores

        S := 0;
        I := 0;

        // inicializacao das condicoes iniciais

        DEN := 1;               // o denominador e uma PA crescente de razao 1
        NUM := 1;               // o numerador e uma potencia de x
        AUX := 0;
        E := 0;         // expoente e uma PA crescente de razao 2
        J := -1;        // o sinal do primeiro termo deve ser positivo

        // recebendo o argumento x do usuario

        leia ( "informe o valor de x: ", X );

        enquanto ( I <= 20 )
        {
                T := NUM / DEN;     // calcula o termo, verificar overflow de DEN
                J := -1 * J;        // inverte o sinal de J
```

```
            S := S + J * T;         // acumula

            // gera os proximos valores da serie

            E := E + 2;             // proximo expoente do argumento X
            NUM := X^E;             // proximo numerador

            // calculo do fatorial do denominador

            DEN := 1;
            AUX := AUX + 2;     // proximo fatorial
            N := AUX;

            repita ateque ( N == 1 )
            {
                  DEN := DEN * N;     // proximo denominador
                  N := N - 1;
            }

            // conta o numero de termos ja calculados

            I := I + 1;
      }
      escreva ( "O valor da soma S e: ", S );
}
```

Problema 1.12.40

Fazer um algoritmo que
a) calcule o valor do co-seno de x através de 20 termos da série seguinte

$$\cos en(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!}$$

b) calcule a diferença entre o valor calculado no item a e o valor fornecido pela função cos(x)
c) imprima o que foi calculado nos itens a e b
observação: o valor de x é fornecido como entrada

```
algoritmo()
{
        // declaracao da estrutura de dados

        declare X:inteiro;          // argumento lido
        declare S:real;             // valor do somatorio
        declare NUM:real;           // o numerador da expressao
        declare DEN:real;           // denominador da expressao
        declare T:real;             // o termo generico da serie
        declare N:inteiro;          // auxiliar no calculo do fatorial
        declare I:inteiro;          // contador do numero de termos
        declare J:inteiro;          // para definir o sinal
        declare E:inteiro;          // expoente de x
        declare K:real;             // valor do cosseno usando a funcao COS
        declare DIF:real;           // a diferenca entre o valor calculado e o
                                    // fornecido pela funcao

        // inicializacao dos acumuladores

        S := 1;         // o primeiro termo e 1
        I := 0;

        // inicializacao das condicoes iniciais

        DEN := 1;           // o denominador e uma PA crescente de razao 2
        NUM := 0;           // o numerador e uma potencia de x
        E := 2;             // expoente e uma PA crescente de razao 2
        J := 1;             // o sinal do primeiro termo deve ser positivo

        // recebendo o argumento x do usuario
        leia ( "informe o valor de x: ", X );

        K := CoSeno(x,"R");
```

```
        enquanto ( I <= 20 )
        {
                E := E + 2;             // proximo expoente do argumento X
                NUM := X^E;             // proximo numerador

                // calculo do fatorial do denominador

                DEN := 1;
                N := E;

                repita ateque ( N == 1 )
                {
                        DEN := DEN * N;             // proximo denominador
                        N := N - 1;
                }

                T := NUM / DEN;     // calcula o termo, verificar overflow de DEN
                J := -1 * J;            // inverte o sinal de J
                S := S + J * T;         // acumula

                // conta o numero de termos ja calculados

                I := I + 1;
        }

        // calculo da diferenca entre K e S

        DIF := K - S;
        DIF := ValorAbs(DIF);
        escreva ( "O valor do co-seno calculado e: ", S );
        escreva ( "O valor do co-seno dado pela formula e: ", K );
        escreva ( "A diferenca e: ", DIF );
}
```

Problemas de Aplicação em Ciências Exatas

Problema 1.12.41

Escrever um algoritmo que
- leia varias linhas, cada uma delas contendo um valor a ser armazenado em x
- para cada valor lido, calcule o valor de y dado pela fórmula

y = 2.5 * cos(x/2)
- escreva os valores de X e Y

observação: A última linha de dados, cujo conteúdo não será processado deverá conter um valor negativo. Use esta condição para testar o fim do processamento

```
algoritmo()
{
	// declaracao das variaveis

	declare X:real;		// argumento lido da entrada
	declare Y:real;		// valor calculado pela funcao dada
	declare C:real;		// valor do cosseno de x
	declare K:real;		// auxiliar na expressao x/2

	enquanto ( X >= 0 )
	{
		// recebendo o argumento x do usuario

		leia ( "informe o valor de x (-1 para encerrar): ", X );

		se ( x >= 0 )
		{
			K := X/2;		// prepara o argumento para a funcao
		/*
		calcula o cosseno de K em radianos utilizando a funcao
		CoSeno() disponivel na biblioteca interna do interpretador
			*/

		/* chama a funcao especificando que o argumento K esta sendo
			passado em radianos
			*/

			C := CoSeno(K,"R");

			Y := 2.5 * C;		// calcula a expressao

			// exibe o resultado

			escreva ( "valor de x: ", X, " valor de Y: ", Y );
```

```
                }
        }
}
```

Problema 1.12.41a

Escrever um algoritmo que
leia várias linhas, cada uma delas contendo um valor a ser armazenado em x
para cada valor lido, calcule o valor de y dado pela fórmula
y = 2.5 * cos(x/2)
escreva os valores de X e Y
observação: A última linha de dados, cujo conteúdo não será processado deverá conter um valor negativo. Use esta condição para testar o fim do processamento

```
algoritmo()
{
	// declaracao das variaveis

	declare X:real;		// argumento lido da entrada
	declare Y:real;		// valor calculado pela funcao dada
	declare C:real;		// valor do cosseno de x

	enquanto ( X >= 0 )
	{
		// recebendo o argumento x do usuario

		leia ( "informe o valor de x (-1 para encerrar): ", X );

		se ( x >= 0 )
		{
			/*
			calcula o cosseno de X em radianos utilizando a funcao
			CoSeno() disponivel na biblioteca interna do interpretador
				*/

			/* chama a funcao especificando que o argumento X esta sendo
				passado em radianos		*/

			C := CoSeno ( X/2,"R" );
			Y := 2.5 * C;		// calcula a expressao

			// exibe o resultado

			escreva ( "valor de x: ", X, " valor de Y: ", Y );
		}
	}
}
```

Problema 1.12.42

Sejam P(x1,y1) e Q(x2,y2) dois pontos quaisquer do plano. A sua distância é dada por

$$d = \sqrt{(x_2^2 - x_1^2)^2 + (y_2^2 - y_1^2)^2}$$

Escrever então um algoritmo que, lendo várias linhas onde cada uma contem as coordenadas dos dois pontos, escreva para cada par de pontos lidos a distância. A última linha contem as coordenadas x1,x2,y1,y2 iguais a zero

```
algoritmo()
{
        // declaracao das variaveis

        declare X1:real;            // abscissa do ponto P
        declare Y1:real;            // ordenada do ponto P
        declare X2:real;            // abscissa do ponto Q
        declare Y2:real;            // ordenada do ponto Q
        declare D:real;             // distancia entre P e Q
        declare Z:inteiro;          // controla o laco enquanto

        // incializa condicao de contorno

        Z := 1;

        enquanto ( Z <> 0 )
        {
                // recebendo as coordenadas

                escreva ( "informe 0 para todas as coordenadas para encerrar");

                leia ( "informe o valor de x1: ", X1 );
                leia ( "informe o valor de Y1: ", Y1 );
                leia ( "informe o valor de X2: ", X2 );
                leia ( "informe o valor de Y2: ", Y2 );

                se ( X1 == 0 & Y1 == 0 ) & ( X2 == 0 & Y2 == 0 )
                {
                        interrompa;
                }

                // calculo da distancia D

                D := Raiz ( Potencia(X2-X1,2) + Potencia(Y2-Y1,2));

                escreva ( "A distancia calculada e: ", D );
```

```
        }
}
```

Problema 1.12.42a

Sejam P(x1,y1) e Q(x2,y2) dois pontos quaisquer do plano. A sua distância é dada por

$$d = \sqrt{(x_2^2 - x_1^2)^2 + (y_2^2 - y_1^2)^2}$$

Escrever então um algoritmo que, lendo várias linhas onde cada uma contem as coordenadas dos dois pontos, escreva para cada par de pontos lidos a distância. A última linha contem as coordenadas x1,x2,y1,y2 iguais a zero

```
algoritmo()
{
        // declaracao das variaveis

        declare X1:real;            // abscissa do ponto P
        declare Y1:real;            // ordenada do ponto P
        declare X2:real;            // abscissa do ponto Q
        declare Y2:real;            // ordenada do ponto Q
        declare D:real;             // distancia entre P e Q
        declare Z:inteiro;          // controla o laco enquanto

        // incializa condicao de contorno

        Z := 1;

        enquanto ( Z <> 0 )
        {
                // recebendo as coordenadas

                escreva ( "informe 0 para todas as coordenadas para encerrar");
                leia ( "informe o valor de x1: ", X1 );
                leia ( "informe o valor de Y1: ", Y1 );
                leia ( "informe o valor de X2: ", X2 );
                leia ( "informe o valor de Y2: ", Y2 );

                se ( X1 == 0 & Y1 == 0 ) & ( X2 == 0 & Y2 == 0 )
                {
                        interrompa;
                }

                // calculo da distancia D

                D := Potencia ( Potencia(X2-X1,2) + Potencia(Y2-Y1,2), 1/2 );

                escreva ( "A distancia calculada e: ", D );
```

```
        }
}
```

Problema 1.12.43

A solução x y para o sistema de equações lineares abaixo:
ax + by = u
cx + dy = v
é dada por x = du - bv / ad - bc e y = av - uc / ad - bc
Escrever então um algoritmo que:

- leia várias linhas, onde cada uma contem os parâmetros a, b, c, d, u, v do sistema (a última linha contem os valores de a, b, c, d iguais a zero)
- calcule a solução x, y de cada sistema dado por seus parâmetros
- escreva os parâmetros lidos e os valores calculados

```
algoritmo()
{
        // declaracao das variaveis

        declare X:real;             // primeira solucao do sistema
        declare Y:real;             // segunda solucao do sistema
        declare A:real;             // coeficiente de x
        declare B:real;             // coeficiente de y
        declare C:real;             // coeficiente de x
        declare D:real;             // coeficiente de y
        declare U:real;             // constante
        declare V:real;             // constante
        declare DEN:real;           // denominador da expressao
        declare Z:inteiro;          // condicao de contorno

        // incializa condicao de contorno

        Z := 1;

        enquanto ( Z <> 0 )
        {
                // recebendo as coordenadas

                escreva ( "Para encerrar informe 0 para todas as coordenadas");
                leia ( "informe o valor do coeficiente A: ", A );
                leia ( "informe o valor do coeficiente B: ", B );
                leia ( "informe o valor do coeficiente C: ", C );
                leia ( "informe o valor de coeficiente D: ", D );
                se ( A == 0 & B == 0 ) & ( C == 0 & D == 0 )
                {
                        interrompa;
                }

                // calculo do denominador
```

```
            DEN := ( A*D - B*C );

            se ( DEN == 0 )
            {
                  escreva ( "erro, denominador nulo" );
            }
            senao
            {
                  // recebe as constantes

                  leia ( "informe o valor da constante U: ", U );
                  leia ( "informe o valor da constante V: ", V );

                  // calculo da solucao X

                  X := ( D*U - B*V ) / DEN;
                  Y := ( A*V - C*U ) / DEN;

                  // resultados

                  escreva ( "Coeficientes A,B,C,D: ", A," ",B," ",C," ",D );
                  escreva ( "Constantes U,V: ", U," ", V );
                  escreva ( "Solucao X e Y: ", X, " ", Y );
            }
      }
}
```

Problema 1.12.44

Fazer um algoritmo que, lendo em uma unidade de entrada os parâmetros A e B de uma reta no plano dado pela equação Y = AX + B, determine a área do triângulo formado por esta reta e os eixos coordenados
O algoritmo lerá um número indeterminado de linhas, cada linha contendo um par de parâmetros (A,B) e para cada par lido deverá escrever os parâmetros A e B e a área do triângulo

```
algoritmo()
{
	// declaracao das variaveis

	declare A:real;		// coeficiente de X na equacao
	declare B:real;		// termo independente na equacao
	declare X:real;		// variavel independente da equacao
	declare Y:real;		// variavel dependente
	declare AREA:real;	// area do triangulo formado
	declare Z:inteiro;	// condicao de contorno

	// incializa condicao de contorno

	Z := 1;

	enquanto ( Z <> 0 )
	{
		// recebendo os parametros

		escreva ( "Para encerrar informe 0 para o coeficiente A");

		leia ( "informe o valor do coeficiente A: ", A );
		leia ( "informe o valor do coeficiente B: ", B );

		se ( A == 0 | B == 0 )
		{
			escreva ( "Coeficientes A,B e Area: ", A, ", ", B, ", ", "0" );
			interrompa;
		}

		/* consideracoes

			Na intersecao da reta Y = AX + B com os eixos coordenados
		ocorre as seguintes propriedades
			X = 0 -> Y = B		intersecao com o eixo X
			Y = 0 -> X = -B/A	intersecao com o eixo Y
```

```
            desse modo a base e a altura do triangulo sao dados pelos valores
            B e B/A        */

            // calculo da area do triangulo

            X := B/A;
            Y := B;
            AREA := X * Y / 2;

            escreva ( "Coeficientes A,B e Area: ", A, ", ", B, " , ", AREA );
        }
}
```

Problema 1.12.45

Fazer um algoritmo para tabular a função y = f(x) + g(x), para x = 1, 2, 3, 4, ... 10 onde:
$h(x) = x^2 - 16$,
f(x) = h(x) se h(x) >= 0 e f(x) = 1 se h(x) < 0,
$g(x) = x^2 + 16$ se f(x) = 0 e g(x) = 0 se f(x) > 0

```
algoritmo()
{
        // declaracao das variaveis

        declare Y:real;             // a funcao dada
        declare X:inteiro;          // a variavel independente
        declare Fx:real;            // o valor da funcao f(x)
        declare Gx:real;            // o valor da funcao g(x)
        declare Hx:real;            // o valor da funcao h(x)
        declare AREA:real;          // area do triangulo formado
        declare Z:inteiro;          // condicao de contorno

        para ( X := 1 ate 10 passo 1 )
        {
                Hx := Potencia(X,2) - 16;           // calculo da funcao h(x)

                se ( Hx >= 0 )
                {
                        Fx := Hx;                   // f(x) = h(x) se h(x) >= 0
                }
                senao
                {
                        Fx := 1;                    // f(x) = 1 se h(x) < 0
                }

                se ( Fx == 0 )
                {
                        Gx := Potencia(X,2) + 16;           // g(x) = x^2 + 16 se f(x) = 0
                }
                senao
                {
                        Gx := 0;                    // g(x) = 0 se f(x) > 0
                }

                // calculo de Y
                Y := Fx + Gx;
```

```
                escreva ( "para x = ", X, " Y = ", Y );
        }
}
```

Problema 1.12.46

As coordenadas de um ponto (x,y) estão disponíveis em uma unidade de entrada. Ler esses valores (até quando um flag ocorrer) e escrever "INTERIOR" se o ponto estiver dentro da região hachurada mostrada abaixo, caso contrário, escrever "EXTERIOR".

```
algoritmo()
{
	// declaracao das variaveis

	declare Y:real;		// a funcao dada
	declare X:inteiro;	// a variavel independente

	// inicializacao da condicao de contorno

	X := 1;
	Y := 1;

	repita ateque ( X == 0 & Y == 0 )
	{
		// recebendo as coordenadas do ponto

		leia ( "informe a coordenada X: ", X );
		leia ( "informe a coordenada Y: ", Y );

		se ( X > 0 )
		{
			// verifica o quadrante 1

			se ( Y < 3*X & Y > X/3 )
			{
				escreva ( "INTERIOR" );
			}
			senao
			{
				escreva ( "EXTERIOR" );
			}
		}
		senao
		{
			se ( X < 0 )
			{
				// verifica o quadrante 3
```

```
                        se ( Y > 3*X & Y < X / 3 )
                        {
                                escreva ( "INTERIOR" );
                        }
                        senao
                        {
                                escreva ( "EXTERIOR" );
                        }
                }
        }
}
```

Comentário: Condição de interioridade do ponto

Pelo diagrama e pelas equações das retas dadas, observamos que as mesmas passam pela origem de coordenadas e notamos também que a região hachurada divide-se em duas:
acima do eixo x e abaixo dele, ocupando o 1º e 3º quadrante. No 1º quadrante x é sempre positivo e no 3º quadrante x é sempre negativo, sendo estas, as condições que devemos analisar com as equações das retas para verificarmos se os pontos cujas coordenadas são lidas estão no interior da região delimitada pelas equações.

No 1º Quadrante observamos que a equação y = 3x estabelece o limite superior de coordenadas, isto é, qualquer ponto com coordenadas y = 3x estará sobre a reta e, qualquer ponto com coordenadas y < 3x estará abaixo e portanto dentro da região hachurada. Por outro lado, a equação y = x/3 estabelece o limite inferior de coordenadas, isto e, qualquer ponto com coordenadas tais que y > x/3 estará acima da reta e portanto dentro da região hachurada.

Dessa forma, podemos escrever que, para o 1º Quadrante, isto é, para X > 0 temos,
da equação y = 3x -> y < 3x
da equação y = x/3 -> y > x/3 então, juntando as duas condições temos que, (x/3 < y < 3x)

No 3º Quadrante, temos que a reta y = 3x é o limite inferior de coordenadas isto é, qualquer ponto, para que o mesmo esteja dentro da área hachurada deverá ter coordenadas tais que y > 3x, pois x agora é negativo. Da mesma forma, a equação y = x/3 define o limite superior da área hachurada e qualquer ponto cujas coordenadas são tais que y < x/3, estará abaixo e portanto dentro da região hachurada.

Desse modo, podemos escrever que para o 3º Quadrante, isto é, para X < 0 temos,
da equação y = 3x -> y > 3x
da equação y = x/3 -> y < x/3

Estas condições são as condições de interioridade dos pontos

Problema 1.12.47

Fazer um algoritmo para calcular e escrever a soma dos cubos dos números pares compreendidos entre A e B. Suponha que os valores de A e B são dados em uma linha e que B > A.

```
algoritmo()
{
	// declaracao das variaveis

	declare A:inteiro;		// o limite inferior dos numeros
	declare B:inteiro;		// o limite superior dos numeros
	declare S:inteiro;		// a soma dos pares entre A e B acima
	declare N:inteiro;		// um numero inteiro qualquer entre A e B
	declare R:inteiro;		// o resto da divisao de N por 2

	// inicializacao dos acumuladores

	S := 0;

	// lendo os limites da sequencia de numeros

	leia ( "informe o limite inferior A: ", A );
	leia ( "informe o limite superior B: ", B );

	N := A;

	enquanto ( N <= B )
	{
		R := Resto ( N, 2 );
		se ( R == 0 )
		{
			// N e par

			R := Potencia ( N, 3 );		// aproveita a variavel R
			S := S + R;
		}
		N := N + 1;
	}
	// resultados
	escreva ( "A soma dos cubos dos pares e: ", S );
}
```

Problema 1.12.48

Fazer um algoritmo que calcule o volume de uma esfera em função do raio R. O raio deverá variar de 0 a 20 cm de 0.5 em 0.5. V = 4 x 3.1415 x $R^3/3$

```
algoritmo()
{
        // declaracao das variaveis

        declare V:real;                 // o volume calculado
        declare R:real;                 // o raio

        R := 0;

        enquanto ( R <= 20 )
        {
                V := 4 * 3.141592 * Potencia ( R, 3 ) / 3;
                escreva ( "R = ", R, " V = ", V );
                R := R + 0.5;
        }
}
```

Problema 1.12.49

Fazer um algoritmo para calcular e escrever a área de um polígono regular de N lados inscrito numa circunferência de raio R. O número de polígonos será fornecido na primeira linha de dados e nas linhas seguintes serão fornecidos os valores de N e R.

```
algoritmo()
{
        // declaracao das variaveis

        declare P:inteiro;          // o numero de poligonos
        declare N:inteiro;          // o numero de lados do poligono
        declare R:real;             // o raio da circunferencia
        declare A:real;             // area do poligono regular inscrito

        // recebe o numero de poligonos

        leia ( "informe o numero de poligonos: ", P );

        // calcula a area dos poligonos inscritos

        enquanto ( P > 0 )
        {
                leia ( "informe o numero de lados: ", N );
                leia ( "informe o raio da circunferencia: ", R );

                // calculo da area

                // A :=
                escreva ( "A area calculada e: ", A );
                P := P - 1;
        }
}
```

Comentário

Polígono regular é aquele que tem todos os lados e todos os ângulos respectivamente congruentes, e portanto inscritíveis em circunferências. O Apótema, representado por m, é distância do lado ao centro do polígono regular. Chamando de p a medida do perímetro do poíigono regular, sua área é dada por A = p.m / 2.

Problema 1.12.50

Para um polígono regular inscrito numa circunferência, quanto maior o número de lados do polígono, mais seu perímetro se aproxima do comprimento da

circunferência. Se o número de lados for muito grande e o raio da circunferência for unitário, o semiperímetro do polígono terá um valor muito proximo de PI. Fazer um algoritmo que escreva uma tabela do semiperímetro em função do número de lados, para polígonos regulares inscritos numa circunferência de raio unitário. O número de lados deverá variar de 5 a 100 de 5 em 5.

```
algoritmo()
{
        // declaracao das variaveis

        declare P:real;                 // medida do perimetro
        declare L:inteiro;              // o numero de lados do poligono

        // inicializacao das condicoes iniciais

        L := 5;

        enquanto ( L < 100 )
        {
                // calculo do perimetro

                // P :=

                escreva ( "Numero de lados = ", L, " Perimetro = ", P );
                L := L + 5;
        }
}
```

Comentário

Polígono regular é aquele que tem todos os lados e todos os ângulos respectivamente congruentes, e portanto inscritíveis em circunferências. O Apótema, representado por m, é distância do lado ao centro do polígono regular. Chamando de p a medida do peíimetro do polígono regular, sua área é dada por A = p.m / 2.

Problema 1.12.51

Construir uma tabela de perda de carga em tubulações para vazões que variem de 0,1 l/s até 10, de 0,1 em 0,1, através da fórmula de Hanzen-Willians dada abaixo:

$J = Q^{1,85} . 10,643.D^{4,87} . C^{-1,85}$
onde:

J = perda de carga (m/1000m)
Q = vazão (m^3/s)
D = diâmetro de tubo (m^2)
C = coeficiente de rugosidade
Os valores de D e C serão lidos de uma unidade de entrada. Considerar como flag o valor D = 0.

```
algoritmo()
{
        // declaracao das variaveis

        declare J: real;        // perda de carga (m/1000m)
        declare Q: real;        // vazao (mü/s)
        declare D: real;        // diametro de tubo (mý)
        declare C: real;        // coeficiente de rugosidade

        // inicializacao das condicoes iniciais

        Q := 0.1;

        // inicializacao da condicao de contorno

        D := 1;

        enquanto ( D <> 0 )
        {
                // recebendo os valores de D e C

                leia ( "informe o diametro do tubo (mý): ", D );

                se ( D <> 0 )
                {
                        leia ( "informe o coeficiente de rugosidade: ", C );

                        // calculo das perdas em funcao da vazao
                        para ( Q := 0.1 ate 10 passo 0.1 )
                        {
                                J := Potencia(Q,1.85) * 10.643*Potencia(D,4.87) *
Potencia(C,-1.85);
                                escreva ( "Q = ", Q, " Perda = ", J );
                        }
                }
        }
}
```

Problema 1.12.51

Fazer um algoritmo que calcule e escreva o número de grãos de milho que se pode colocar num tabuleiro de xadrez, colocando 1 no primeiro quadro e nos quadros seguintes o dobro do quadro anterior

$S = 2^0 + 2^1 + 2^2 + 2^3 + ... + 2^{64}$

```
algoritmo()
{
        // declaracao das variaveis

        declare S: real;              // o numero de graos de milho colocados
        declare N: inteiro;           // expoente de 2

        // inicializacao das condicoes iniciais

        N := 0;

        enquanto ( N <= 64 )
        {
                // acumula

                S := S + Potencia(2,N);
                N := N + 2;
        }
        escreva ( "A quantidade de milho colocada e: ", S );
}
```

Problema 1.12.53

Um certo aço é classificado de acordo com o resultado de três testes que devem verificar se o mesmo satisfaz as seguintes especificações:

- teste 1 - conteúdo de carbono abaixo de 7%
- teste 2 - dureza Rokwell maior que 50
- teste 3 - resistência a tração maior do que 80.000 psi

Ao aço é atribuido o grau 10, se passa pelos três testes, 9, se passa apenas nos testes 1 e 2, 8, se passa no teste 1 e 7 se não passou nos três testes. Supondo que sejam lidos de uma unidade de entrada: o número da amostra, conteúdo de carbono em %, a dureza rokwell e a resistência a tração, fazer um algoritmo que de a classificação de 112 amostras de aço que foram testadas, escrevendo o número da amostra e o grau obtido.

```
algoritmo()
{
	// declaracao das variaveis

	declare AMOSTRA: inteiro;	// o numero da amostra
	declare CARBONO: real;		// o percentual de carbono
	declare DUREZA: real;		// a dureza rokwell
	declare RESISTENCIA: real;	// resistencia a tracao
	declare I: inteiro;		// contador do numero de amostras
	declare GRAU:inteiro;		// resultado dos testes

	// inicializacao dos contadores

	I := 1;

	enquanto ( I <= 112 )
	{
		// recebe os parametros da amostra

		leia ( "informe o numero da amostra: ", AMOSTRA );
		leia ( "informe o % de carbono: ", CARBONO );
		leia ( "informe a dureza Rokwell: ", DUREZA );
		leia ( "informe a resistencia a tracao: ", RESISTENCIA );

		// verifica valores para definir o grau da amostra

		se ( CARBONO < 7 & DUREZA > 50 ) & ( RESISTENCIA > 80000 )
		{
			GRAU := 10;
		}
		senao
		{
```

```
                    se ( CARBONO < 7 & DUREZA > 50 )
                    {
                        GRAU := 9;
                    }
                    senao
                    {
                        se ( CARBONO < 7 )
                        {
                            GRAU := 8;
                        }
                        senao
                        {
                            GRAU := 7;
                        }
                    }
                }

                // resultado

                escreva ( "A amostra ", AMOSTRA, " obteve grau: ", GRAU );

                // acrescenta 1 ao contador

                I := I + 1;
            }
}
```

Problema 1.12.54

Fazer um algoritmo para calcular a raiz quadrada de um número positivo usando o roteiro abaixo, baseado no método de aproximações sucessivas de Newton. Seja Y o número:

- a primeira aproximação para a raiz quadrada de y e x = y/2
- as sucessivas aproximações serao $X_{n+1} = X_n^2 + Y / 2X_n$

O algoritmo deverá prever 20 aproximações

```
algoritmo()
{
        // declaracao das variaveis

        declare Y: real;            // o numero dado
        declare X1: real;           // aproximacoes
        declare X2: real;           // aproximacoes
        declare I: inteiro;         // contador das aproximacoes

        // recebe o numero Y para calcular a raiz quadrada

        leia ( "informe um numero positivo: ", Y );

        se ( Y > 0 )
        {
                // inicializacao das condicoes iniciais, primeira aproximacao

                X1 := Y / 2;
                I := 0;

                // faz o calculo com 20 aproximacoes

                repita ateque ( I == 20 )
                {
                        X2 := ( Potencia(X1,2) + Y ) / ( 2 * X1 );
                        X1 := X2;

                        // incrementa o contador em 1

                        I := I + 1;
                }

                // resultado
                escreva ( "A raiz quadrada de ", Y, " e ", X2 );
        }
        senao
```

```
        {
                escreva ( "informe um numero positivo" );
        }
}
```

Problema 1.12.55

Dada a equação $x^3 - 3x^2 + 1 = 0$, pode-se encontrar qualquer uma de suas raizes reais através de aproximações sucessivas utilizando a seguinte fórmula:
$X_b = X_a - X_a^3 - 3X_a^2 + 1 / 3X_a^2 - 6x_a$
Fazer um algoritmo que:

- considere como primeira aproximação x = 1,5
- calcule e escreva a trigesima aproximação da raiz

```
algoritmo()
{
        // declaracao das variaveis

        declare Xa: real;           // aproximacoes
        declare Xb: real;           // aproximacoes
        declare I: inteiro;         // contador das aproximacoes
        declare Y: real;            // o valor da funcao para a raiz calculada

        // atribuicao das condicoes iniciais

        Xa := 1.5;

        I := 0;

        // faz o calculo com 30 aproximacoes

        repita ateque ( I == 30 )
        {
                Xb := Xa - ( Potencia(Xa,3) - (3*Potencia(Xa,2)) + 1 ) /
                        ( (3*Potencia(Xa,2)) - 6*Xa);
                Xa := Xb;

                // incrementa o contador em 1

                I := I + 1;
        }

        // resultado

        escreva ( "A trigesima aproximacao e ", Xb );

        // substituindo o valor de Xb na funcao temos
        Y := Potencia(Xb,3) - 3*Potencia(Xb,2) + 1;

        escreva ( "o valor da funcao e: ", Y );
```

```
}
```

Problema 1.12.56

Fazer um algoritmo que tabule a seguinte função
$f(x,y) = x^2 + 3x + y^2 / xy - 5y - 3x + 15$
para x = 1, 4, 9, 16, ... 100 e
para y = 0, 1, 2, ... 5 para cada valor de x

```
algoritmo()
{
	// declaracao das variaveis

	declare F: real;		// valor da funcao f(x,y) calculado
	declare X: real;		// valor da variavel x
	declare Y: real;		// valor da variavel y
	declare D: real;		// denominador da expressao

	para ( X := 1 ate 10 passo 1 )
	{
		para ( Y := 1 ate 5 passo 1 )
		{
			// calculo do denominador

			D := X*Y - 5*Y + 15;

			se ( D <> 0 )
			{
				// calcula a funcao f(x,y)

				F := (Potencia(X,2) + 3*X + Potencia(Y,2)) / D;
				escreva ( "x = ", X, " y = ", Y, " f(x,y) = ", F );
			}
			senao
			{
				escreva ( "Denominador nulo para x = ", X, " e y = ", Y );
			}
		}
	}
}
```

Problema 1.12.57

Tem-se 10 conjuntos de valores, onde cada conjunto é formado pelo número de um aluno, a nota provisória do seu trabalho prático e a data em que foi entregue. Fazer um algoritmo para:
a) calcular e imprimir a nota final de cada aluno, sabendo-se que os trabalhos entregues:
até 20/04, nota final = nota provisória + 10
até 02/05, nota final = nota provisória
até 30/05, nota final = nota provisória / 2;
até 30/06, nota final = 0
b) calcular a média e o desvio padrão das notas provisória e final
Observacao: consultar fórmula do desvio padrao

$$desvio = \sqrt{\frac{1}{N-1}\left[\sum_{i=1}^{N} x_i^2 - \frac{1}{N}\left(\sum_{i-1}^{N} x_i\right)^2\right]}$$

```
algoritmo()
{
        // declaracao das variaveis

        declare N: inteiro;          // numero de conjuntos de dados
        declare I: inteiro;          // contador da quantidade conjuntos
        declare ALUNO: inteiro;      // numero do aluno
        declare PROV: real;          // valor da nota provisoria
        declare DIA: inteiro;        // dia em que o trabalho foi entregue
        declare MES: inteiro;        // mes em que o trabalho foi entregue
        declare FINAL: real;         // nota final do aluno
        declare MEDIAP: real;        // media das notas provisorias
        declare MEDIAF: real;        // media das notas finais
        declare SOMAP: real;         // soma das notas provisorias
        declare SOMAF: real;         // soma das notas finais
        declare DESVIOP: real;       // desvio padrao das notas provisorias
        declare DESVIOF: real;       // desvio padrao das notas finais
        declare SOMAQP: real;        // somatorio dos quadrados das notas provisorias
        declare SOMAQF: real;        // somatorio dos quadrados das notas finais

        // inicializacao dos acumuladores

        SOMAP := 0;
        SOMAF := 0;

        // incializacao dos contadores

        I := 1;
```

```
// inicializacao das condicoes iniciais

N := 10;

enquanto ( I <= 10 )
{
        // recebe o conjunto de dados de cada aluno

        leia ( "informe o numero do aluno: ", ALUNO );
        leia ( "informe a nota provisoria: ", PROV );
        leia ( "informe o dia da entrega: ", DIA );
        leia ( "informe o mes da entrega: ", MES );

        // calculo da nota final

        se ( MES == 4 )
        {
                se ( DIA <= 20 )
                {
                        FINAL := PROV + 10;
                }
                senao
                {
                        FINAL := PROV;
                }
        }
        senao
        {
                se ( MES == 5 )
                {
                        se ( DIA <= 2 )
                        {
                                FINAL := PROV;
                        }
                        senao
                        {
                                FINAL := PROV / 2;
                        }
                }
                senao
                {
                        FINAL := 0;
                }
        }
```

```
            // resultado de cada aluno

            escreva ( "A nota final do aluno: ", ALUNO, " e ", FINAL );

            // acumula as notas provisorias e finais

            SOMAP := SOMAP + PROV;
            SOMAF := SOMAF + FINAL;

            // calculo da soma dos quadrados de cada uma das notas

            SOMAQP := SOMAQP + Potencia(PROV,2);
            SOMAQF := SOMAQF + Potencia(FINAL,2);

            // conta o aluno

            I := I + 1;
        }

        // calculo da media

        MEDIAP := SOMAP / N;
        MEDIAF := SOMAF / N;

        // calculo do desvio padrao

        DESVIOP := ( SOMAQP - ( Potencia(SOMAP,2) ) / N ) / ( N - 1 );
        DESVIOP := Raiz ( DESVIOP );

        DESVIOF := ( SOMAQF - ( Potencia(SOMAF,2) ) / N ) / ( N - 1 );
        DESVIOF := Raiz ( DESVIOF );

        // resultados

        escreva ( "Media das notas provisorias: ", MEDIAP );
        escreva ( "Media das notas finais: ", MEDIAF );
        escreva ( "Desvio padrao das notas provisorias: ", DESVIOP );
        escreva ( "Desvio padrao das notas finais: ", DESVIOF );
}
```

Problema 1.12.58

Números complexos podem ser escritos na forma cartesiana z = x + yi ou na forma exponencial $z = re^{\theta i}$. Multiplicações e divisões de números complexos na forma exponencial ficam muito mais fáceis de serem feitas, pois assumem a seguinte forma:

$z1 . z2 = (r1 . r2).e^{(\theta 1+\theta 2)i}$

$z1 / z2 = (r1 / r2).e^{(\theta 1-\theta 2)i}$

bastando, portanto, operar os módulos r1 e r2 e os argumentos $\theta 1$ e $\theta 2$. Fazer um algoritmo que leia um conjunto de linhas, cada uma contendo um código e quatro valores. Código MULTIPLICA indica que se quer operar a multiplicação, código DIVIDE indica que se quer operar a divisão e código VAZIO vai indicar fim dos dados. Para cada operação completada, escrever todos os valores lidos e calculados

```
algoritmo()
{
	// declaracao das variaveis

	declare CODIGO: caracter;	// codigo do tipo de operacao - M,D,F
	declare R1: real;		// modulo do primeiro numero complexo
	declare R2: real;		// modulo do segundo numero complexo
	declare A1: real;		// valor do argumento de R1
	declare A2: real;		// valor do argumento de r2
	declare R: real;		// resultado da operacao com os modulos
	declare A: real;		// resultado da operacao com os
argumentos

	// incializacao da condicao de contorno

	CODIGO := ' ';

	enquanto ( CODIGO <> 'F' | CODIGO <> 'f' )
	{
		// recebe os modulos, os argumento e a operacao

		leia ( "informe o codigo: ", CODIGO );

		se ( CODIGO == 'F' | CODIGO == 'f' )
		{
			interrompa;
		}

		leia ( "informe o modulo R1: ", R1 );
		leia ( "informe o modulo R2: ", R2 );
		leia ( "informe o argumento A1: ", A1 );
```

```
            leia ( "informe o argumento A2: ", A2 );

            // calculos

            se ( CODIGO == 'M' | CODIGO == 'm' )
            {
                  // multiplicacao

                  R := R1 * R2;
                  A := A1 + A2;
            }
            senao
            {
                  se ( CODIGO == 'D' | CODIGO == 'd' )
                  {
                        // divisao

                        se ( R2 <> 0 )
                        {
                              R := R1 / R2;
                              A := A1 - A2;
                        }
                        senao
                        {
                              escreva ( "Divisao invalida !, R2 e nulo" );
                        }
                  }
            }

            // resultados

            escreva ( "Modulo resultante: ", R );
            escreva ( "Argumento resultante: ", A );
      }
}
```

Problema 1.12.59

O cálculo do valor de uma integral definida, usando o método da aproximação por trapézios é feito dividindo o intervalo de integração em N partes iguais e aproximando a função, em cada subintervalo obtido, por um segmento de reta. O valor da integral é calculado, então, como a soma das áreas dos diversos trapézios formados.
$A = (Y_i + Y_{i+1})h$, área de cada trapézio
$h = X_{i+1} - X_i = b - a / N$, constante
Fazer um algoritmo para determinar e escrever o valor de pi, o qual pode ser calculado pela integral

$$y = \int_0^1 \frac{1}{1+x^2}\,dx$$

```
algoritmo()
{
        // declaracao das variaveis

        declare S: real;            // valor a ser calculado
        declare Y1: real;           // o primeiro valor que a funcao assume
        declare Y2: real;           // o segundo valor que a funcao assume
        declare H: real;            // base do trapezio
        declare a: inteiro;         // limite inferior de integracao
        declare b: inteiro;         // limite superior de integracao
        declare X: real;            // a variavel independente
        declare N: inteiro;         // numero de particoes do intervalo de integracao
        declare I: inteiro;         // contador do numero de trapezios ja somados

        // definicao das condicoes inciais

        a := 0;
        b := 1;
        N := 100;    // valor arbitrariamente definido
        X := a;

        // inicializa os acumuladores

        S := 0;

        // calculo do incremento de x, a base do trapezio

        H := ( b - a ) / N;

        // inicializacao da condicao de contorno

        I := 1;
```

```
        repita ateque ( I == N | X > b )
        {
                // calculo de Y

                Y1 := 1 / ( 1 + Potencia(X,2));
                Y2 := 1 / ( 1 + Potencia(X+H,2));

                // acumula a area S (valor de pi)

                S := S + ( Y1 + Y2 ) * H / 2;

                // conta o numero de trapezios ou de particoes

                I := I + 1;

                // incrementa X para o proximo valor do intervalo [a,b]

                X := X + H;
        }

        // resultado

        escreva ( "O valor calculado de S e ", S );
}
```

Comentário:

A área do trapézio é base maior mais base menor vezes altura dividido por 2

Problema 1.12.60

Fazer um algoritmo que:

- leia um conjunto de 25 linhas, contendo, cada uma três numeros inteiros positivos (em qualquer ordem)
- calcule o máximo divisor comum entre os três números lidos, utilizando o método das divisões sucessivas
- escreva os três números lidos e o MDC entre eles

```
algoritmo()
{
        // declaracao das variaveis

        declare N: inteiro;                  // numero de linhas com dados
        declare A,B,C,MDC: inteiro;          // tres numeros quaisquer e o MDC
        declare MAI, MED, MEN:inteiro;       // os numeros A,B,C ordenados
        declare Q: inteiro;                  // auxiliar nos calculos
        declare R: inteiro;                  // auxiliar nos calculos

        // definicao das condicoes inciais

        N := 25;              // ha 25 linhas com dados (tres numeros quaisquer)

        repita ateque ( N == 0 )
        {
                leia ( "informe o 1º numero: ", A );
                leia ( "informe o 2º numero: ", B );
                leia ( "informe o 3º numero: ", C );

                // atribuicoes para resguardar valores iguais que nao estao
                // sendo considerados na ordenacao abaixo

                MAI := A;
                MED := B;
                MEN := C;

                // ordena os tres numeros lidos

                se ( A > B & A > C )
                {
                        MAI := A;
                        se ( B > C )
                        {
                                MED := B;
                                MEN := C;
                        }
```

```
                senao
                {
                        MED := C;
                        MEN := B;
                }
        }
        senao
        {
                se ( B > A & B > C )
                {
                        MAI := B;
                        se ( A > C )
                        {
                                MED := A;
                                MEN := C;
                        }
                        senao
                        {
                                MED := C;
                                MEN := A;
                        }
                }
                senao
                {
                        se ( C > A & C > B )
                        {
                                MAI := C;
                                se ( A > B )
                                {
                                        MED := A;
                                        MEN := B;
                                }
                                senao
                                {
                                        MED := B;
                                        MEN := A;
                                }
                        }
                }
        }
        escreva ( "Os numeros ordenados sao: ", MEN, " ", MED, " ", MAI );

        // calcula o MDC entre os numeros, agora, ordenados
        // calculo do MDC entre MAI e MEN
```

```
            repita ateque ( MEN == 0 )
            {
                  Q := ParteInteira ( MAI / MEN );
                  R := MAI - Q*MEN;
                  MAI := MEN;
                  MEN := R;
            }

            MDC := MAI;

            // ordena o resultado com os outros valores

            se ( MED > MDC )
            {
                  MAI := MED;
                  MEN := MDC;
            }
            senao
            {
                  MAI := MDC;
                  MEN := MED;
            }

            // calcula o MDC entre os numeros, agora, com nova ordenacao
            // calculo do MDC entre os novos MAI e MEN

            repita ateque ( MEN == 0 )
            {
                  Q := ParteInteira ( MAI / MEN );
                  R := MAI - Q*MEN;
                  MAI := MEN;
                  MEN := R;
            }

            MDC := MAI;          // este e o MDC final

            // resultados

            escreva ( "Numeros: ", A, ", ", B, ", ", C, " MDC = ", MDC );

            N := N - 1;          // conta uma linha processada
      }
}
```

Problema 1.12.61

O número 3025 possui a seguinte característica
30 + 25 = 55 e 55^2 = 3025
Fazer um algoritmo que pesquise e imprima todos os números de quatro algarismos que apresentam tal característica

```
algoritmo()
{
        // declaracao das variaveis

        declare N: inteiro;             // um numero qualquer de 4 algarismos
        declare PP: inteiro;            // os dois primeiros algarismos
        declare UU: inteiro;            // os dois ultimos algarismos

        // definicao das condicoes inciais

        N := 1000;              // apenas numeros de 4 digitos

        repita ateque ( N > 9999 )
        {
                PP := N / 100;          // pega os dois primeiros digitos de N
                UU := N - PP*100;       // pega os dois ultimos digitos de N

                // testa pela caracteristica

                se ( ( PP + UU )^2 == N )
                {
                        escreva ( "encontrado: ", N );
                }

                N := N + 1;             // proximo numero de 4 digitos
        }
}
```

Problema 1.12.61a

O número 3025 possui a seguinte característica
30 + 25 = 55 e 55^2 = 3025
Fazer um algoritmo que pesquise e imprima todos os números de quatro algarismos que apresentam tal característica

```
algoritmo()
{
	// declaracao das variaveis

	declare N: inteiro;		// um numero qualquer de 4 algarismos
	declare PP: real;		// os dois primeiros algarismos
	declare UU: real;		// os dois ultimos algarismos

	// definicao das condicoes inciais

	N := 1000;		// apenas numeros de 4 digitos

	repita ateque ( N > 9999 )
	{
		PP := ParteInteira (N/100);		// pega os dois primeiros digitos de
N
		UU := ParteInteira (N-PP*100);	// pega os dois ultimos digitos de N

		// testa pela caracteristica

		se ( Potencia(PP+UU,2) == N )
		{
			escreva ( "encontrado: ", N );
		}

		N := N + 1;		// proximo numero de 4 digitos
	}
}
```

Problema 1.12.62

Dada uma equação diferencial y = f(x,y) e a condição inicial y(X0) = Y0, pode-se encontrar uma solução aproximada desta equação, usando o seguinte método:
$y_1 = y_0 + hf(X_0,Y_0)$
$y_2 = y_1 + hf(X_1,Y_1)$
...
$Y_{k+1} = Y_k + hf(X_k,Y_k)$
onde h é um acréscimo que se dá aos valores de x.
$h = X_n - X_0 / n$, onde X_n é o limite superior do intervalo, X_0 é o limite inferior do intervalo e n é o número de subintervalos. Fazer um algoritmo que encontre e escreva as soluções aproximadas da equacao Y' = xy com Y(0) = 1 no intervalo fechado de 0 a 1, com n = 10 subintervalos

```
algoritmo()
{
        // declaracao das variaveis

        declare N: inteiro;          // numero de subintervalos
        declare X0: real;            // limite inferior do intervalo
        declare Xn: real;            // limite superior do intervalo
        declare h: real;             // acrescimo passo a passo
        declare Y0: real;            // valor da primeira aproximacao
        declare Yn: real;            // as sucessivas aproximacoes
        declare I: inteiro;          // contador para o numero de intervalos

        // definicao das condicoes inciais

        N := 10;              // 10 subintervalos
        X0 := 0;
        Xn := 1;

        // calculo do passo de x pela expressao dada

        h := ( Xn - X0 ) / N;

        // condicao inicial de processamento

        Xn := X0;             // primeiro valor do intervalo
        Yn := 1;              // valor da funcao no para X0
        I := 0;

        repita ateque ( I > N )
        {
                Yn := Y0 + h * Xn * Yn;
                Y0 := Yn;             // o proximo e funcao do anterior
                Xn := Xn + h;         // proximo x no intervalo [0,1]
```

```
                I := I + 1;              // conta o numero de intervalos
        }

        // resultado

        escreva ( "A solucao da equacao e : ", Yn );
}
```

Problema 1.12.63

Fazer um algoritmo que:
- calcule o número de divisores dos números compreendidos entre 300 e 400
- escreva cada número e o número de divisores correspondentes

```
algoritmo()
{
	// declaracao das variaveis

	declare N: inteiro;		// um numero qualquer entre 300 e 400
	declare I: inteiro;		// contador para o numero de divisores
	declare Q: real;		// quociente
	declare D: inteiro;		// divisores de N
	declare L: real;		// limite do divisor de N

	// definicao das condicoes inciais

	N := 300;		// valor inicial do intervalo [300,400]

	repita ateque ( N > 400 )
	{
		L := Raiz (N);		// limite superior dos divisores de N

		I := 0;		// o numero de divisores para cada numero e
				// zero inicialmente, pois ainda nao se achou nenhum
		D := 1;		// inicio dos divisores

		repita ateque ( D > L )
		{
			Q := ParteInteira ( N / D );

			// verifica se e divisor exato

			se ( Q * D == N )
			{
				// e divisor exato

				I := I + 1;
			}

			D := D + 1;		// proximo divisor
		}

		// resultados
```

```
            escreva ( "Numero: ", N, " Quantidade de divisores: ", I );
            N := N + 1;              // proximo N ate 400
        }
}
```

Problema 1.12.64

Fazer um algoritmo que, dado 100 números inteiros positivos, calcule e imprima os que são números perfeitos.
Nota: Número perfeito é aquele cuja soma de seus divisores, exceto ele próprio é igual ao número.
Exemplo: 6 é perfeito porque 1 + 2 + 3 = 6

```
algoritmo()
{
	// declaracao das variaveis

	declare N: inteiro;		// um numero qualquer entre os 100
	declare S: inteiro;		// soma dos divisores de N, com excessao de N
	declare Q: real;		// quociente
	declare D: inteiro;		// divisores de N
	declare L: real;		// limite do divisor de N

	// definicao das condicoes inciais

	N := 0;		// valor inicial do intervalo [0,100]
			// definido arbitrariamente, apenas para exemplificar

	repita ateque ( N > 100 )
	{
		L := N / 2;		// limite superior dos divisores de N
					// definido dessa forma para garantir que
					// pelo menos a metade do numero seja testada

		S := 0;			// a soma dos divisores de N

		D := 1;			// inicio dos divisores

		repita ateque ( D > L )
		{
			Q := ParteInteira ( N / D );

			// verifica se e divisor exato

			se ( Q * D == N )
			{
				// e divisor exato
				S := S + D;
			}

			D := D + 1;		// proximo divisor
```

```
            }

            // resultados

            se ( S == N )
            {
                  escreva ( "Numero perfeito: ", N );
            }

            N := N + 1;             // proximo N
      }
}
```

Problema 1.12.65

Regressão linear é uma técnica estatística que ajusta uma equação linear (da forma y = ax + b) a um conjunto de pontos dados. O problema consiste em achar uma equação linear que melhor se ajuste aos pontos dados. Um dos métodos empregados é o dos mínimos quadrados, que consiste em minimizar a soma dos quadrados dos desvios verticais dos pontos para a linha reta. As fórmulas para os coeficientes a e b, dado um conjunto de n pares de pontos (x,y) são:

$$a = \frac{n\sum xy - \sum x.\sum y}{n\sum x^2 - \left(\sum x\right)^2} \qquad b = \frac{\sum y.\sum x^2 - \sum x.\sum xy}{n\sum x^2 - \left(\sum x\right)^2}$$

$$R = \frac{n\sum xy - \sum x.\sum y}{\left(\sqrt{n\sum x^2 - \left(\sum x\right)^2}\right)\left(\sqrt{n\sum y^2 - \left(\sum y\right)^2}\right)}$$

Uma vez achada a equação da reta, é importante determinar a precisão de ajustamento dessa linha aos dados reais. Uma medida disso é o coeficiente de correlação R, dado pela fórmula ...
O intervalo de variação de R é de -1 <= R <= 1. Quanto mais próximo de 1 ou -1 ficar o valor de R, melhor terá sido o ajustamento da reta. Fazer um algoritmo para ler e imprimir um conjunto de pares de pontos (x,y) e calcular e escrever os valores de a, b e R

```
algoritmo()
{
        // declaracao das variaveis

        declare a: real;            // coeficiente de x na equacao y = ax + b
        declare b: real;            // termo independente na equacao y = ax + b
        declare N: inteiro;         // numero de pares de pontos
        declare I: inteiro;         // contador para N
        declare x: real;            // abscissa
        declare y: real;            // ordenada
        declare Sx: real;           // soma dos x
        declare Sy: real;           // soma dos y
        declare SPxy: real;         // soma dos produtos xy
        declare SQx: real;          // soma dos quadrados de x
        declare SQy: real;          // soma dos quadrados de y
        declare R: real;            // coeficiente de correlacao
        declare DENX: real;         // denominador com expressao em x
        declare DENY: real;         // denominador com expressao em y

        // inicializacao dos acumuladores

        Sx := 0;
        Sy := 0;
```

```
SPxy := 0;
SQx := 0;
SQy := 0;

// le o numero de pares a serem recebidos
leia ( "informe o numero de pares: ", N );

// inicializa o contador para N

I := 0;

// le os pares de numeros e faz as devidas acumulacoes

repita ateque ( I == N )
{
        leia ( "informe o valor de x: ", x );
        leia ( "informe o valor de y: ", y );

        // acumula a soma das coordenadas

        Sx := Sx + x;
        Sy := Sy + y;
        SPxy := SPxy + x*y;
        SQx := SQx + Potencia(x,2);
        SQy := SQy + Potencia(y,2);

        // conta o par que foi lido

        I := I + 1;
}

// os denominadores devem ser diferentes de zero

DENX := N*SQx - Potencia(Sx,2);
DENY := N*SQy - Potencia(Sy,2);

se ( DENX <> 0 & DENY <> 0 )
{
        a := ( N*SPxy - Sx*Sy ) / DENX;
        b := ( Sy*Sqx - Sx*SPxy ) / DENX;
        R := ( N*SPxy - Sx*Sy ) / ( Raiz(DENX) * Raiz(DENY) );

        // resultados

        escreva ( "Coeficiente a = ", a , " b = ", b, " e R = ", R );
```

```
        }
        senao
        {
                escreva ( "denominadores nulos" );
        }
}
```

Problema 1.12.66

Capícuas são números que tem o mesmo valor, se lidos da esquerda para a direita ou da direita para a esquerda. Fazer um algoritmo que determine e escreva todos os números inteiros menores que 10.000 que são capícuas e quadrados perfeitos ao mesmo tempo.

```
algoritmo()
{
	// declaracao das variaveis

	declare N: inteiro;		// um numero qualquer entre 0 e 10.000
	declare A,B,C,D: inteiro;	// auxiliares
	declare R: real;		// raiz do numero N

	// definicao das condicoes inciais

	N := 0;		// valor inicial do intervalo [0,10.000]

	// numeros com 1 digito

	repita ateque ( N > 9 )
	{
		// para numeros com apenas 1 digito basta verificar
		// se ele e um quadrado perfeito

		R := Raiz (N);
		R := ParteInteira(R);
		se ( R*R == N )
		{
			escreva ( N );
		}
		N := N + 1;
	}

	// numeros com 2 digitos

	repita ateque ( N > 99 )
	{
		// verifica se e quadrado perfeito

		R := Raiz (N);
		R := ParteInteira(R);
		se ( R*R == N )
		{
			// verifica se e capicua
```

```
                A := ParteInteira ( N / 10 );
                B := ParteInteira ( N - A * 10 );
                se ( A == B )
                {
                        escreva ( N );
                }
        }
        N := N + 1;
}

// numeros com 3 digitos

repita ateque ( N > 999 )
{
        // verifica se e quadrado perfeito

        R := Raiz (N);
        R := ParteInteira(R);
        se ( R*R == N )
        {
                // verifica se e capicua

                A := ParteInteira ( N / 100 );
                B := ParteInteira ( N - A * 100 ) / 10;
                se ( A == B )
                {
                        escreva ( N );
                }
        }
        N := N + 1;
}

// numeros com 4 digitos

repita ateque ( N > 9999 )
{
        // verifica se e quadrado perfeito
        R := Raiz (N);
        R := ParteInteira(R);
        se ( R*R == N )
        {
                // verifica se e capicua

                A := ParteInteira ( N / 1000 );
```

```
                    B := ParteInteira ( N - A * 10000 ) / 100;
                    C := ParteInteira ( N - A*1000 - B*100 ) / 10;
                    D := N - A*1000 - B*100 - C*10;
                    se ( A == D & B == C )
                    {
                        escreva ( N );
                    }
                }
                N := N + 1;
        }
}
```

Problema 1.12.66a

Capícuas são números que tem o mesmo valor, se lidos da esquerda para a direita ou da direita para a esquerda. Fazer um algoritmo que determine e escreva todos os números inteiros menores que 10.000 que são capícuas.

```
algoritmo()
{
	// declaracao das variaveis

	declare N: inteiro;			// um numero qualquer entre 0 e 10.000
	declare A,B,C,D,E: inteiro;		// auxiliares

	// definicao das condicoes inciais

	N := 0;			// valor inicial do intervalo [0,10.000]

	// numeros com 1 digito

	repita ateque ( N > 100000 )
	{
		se ( N <= 9 )
		{
			// todos os numeros com 1 digito sao capicuas

			escreva ( N );
		}
		senao
		{
			se ( N <= 99 )
			{
				// numeros com 2 digitos

				A := ParteInteira ( N / 10 );
				B := ParteInteira ( N - A * 10 );
				se ( A == B )
				{
					escreva ( N );
				}
			}
			senao
			{
				se ( N <= 999 )
				{
					// numeros com 3 digitos
```

```
                                A := ParteInteira ( N / 100 );
                                B := ParteInteira ( N - A * 100 );
                                se ( A == B )
                                {
                                        escreva ( N );
                                }
                        }
                        senao
                        {
                                se ( N <= 9999 )
                                {
                                        // numeros com 4 digitos

                                        A := ParteInteira ( N / 1000 );
                                        B := ParteInteira ( N - A * 1000 ) / 100;
                                        C := ParteInteira ( N - A*1000 - B*100 ) / 10;
                                        D := N - A*1000 - B*100 - C*10;
                                        se ( A == D & B == C )
                                        {
                                                escreva ( N );
                                        }
                                }
                                senao
                                {
                                        // numeros com 5 digitos

                                        A := ParteInteira ( N / 10000 );
                                        B := ParteInteira ( N - A * 10000 ) / 1000;
                                        C := ParteInteira ( N - A*10000 - B*1000 )
/ 100;
                                        D := ParteInteira ( N - A*10000 - B*1000 -
C*100 ) / 10;
                                        E := N - A*10000 - B*1000 - C*100 - D*10;
                                        se ( A == E & B == D )
                                        {
                                                escreva ( N );
                                        }
                                }
                        }
                }
        }
        N := N + 1;
    }
}
```

Problema 1.12.67

Número primo é aquele que só é divisível por ele mesmo e pela unidade. Fazer um algoritmo que determine e escreva os números primos compreendidos entre 5.000 e 7.000.

```
algoritmo()
{
	// declaracao das variaveis

	declare N: inteiro;		// um numero inteiro entre 5.000 e 7.000
	declare D: inteiro;		// possiveis divisores de N
	declare R: real;		// quociente auxiliar
	declare P: inteiro;		// indica se um numero e primo ou não

	// definicao das condicoes inciais

	N := 5001;		// valor inicial do intervalo [5000,7000]

	repita ateque ( N > 7000 )
	{
		// inicializacao dos divisores para cada numero N

		D := 3;		// os numeros pares nao serao testados

		P = 1;		// indica que e primo

		repita ateque ( D > N / 2 )
		{
			// a metade do numero N e o limite superior de teste pois
			// e o maior valor que divide N

			// verifica se D e divisor de N

			R := Resto ( N, D );
			se ( R == 0 )
			{
				// D e divisor, interrompe o laco

				P := 0;		// indica que nao e primo
				interrompa;
			}
			D := D + 2;		// proximo divisor, apenas os impares
		}
```

```
                se ( P == 1 )
                {
                        // o numero N e primo

                        escreva (N);
                }

                N := N + 2;             // testar apenas os numeros impares
        }
}
```

Problema 1.12.68

Fazer um algoritmo que:

- leia um conjunto de linhas contendo, cada uma, um numero inteiro, na base 10, de até 5 dígitos. A última linha contem o valor zero.
- transforme esse número da base 10 para a base 2
- escreva o número na base 10 e na base 2

```
algoritmo()
{
	// declaracao das variaveis

	declare N: inteiro;		// um numero inteiro qulaquer de ate 5 digitos
	declare R: inteiro;		// resto das divisoes de N por 2
	declare AUX: inteiro;		// uma variavel auxiliar cujo valor inicial e N

	// incializacao da condicao de contorno

	N := 1;

	repita ateque ( N == 0 )
	{
		// leitura do numero N
		leia ( "informe um numero inteiro: ", N );
		se ( N <> 0 )
		{
			AUX := N;	// preservar o valor de N que deve ser mostrado
			escreva ( "Base 2:" );
			repita ateque ( AUX == 0 )
			{
				R := Resto ( AUX, 2 );
				escreva ( R );
				AUX := AUX / 2;
			}
			escreva ( "Base 10: ", N );
		}
	}
}
```

Problema 1.12.68a

Fazer um algoritmo que:
- leia um conjunto de linhas contendo, cada uma, um número inteiro, na base 10, de até 5 dígitos. A última linha contem o valor zero.
- transforme esse numero da base 10 para a base 2
- escreva o numero na base 10 e na base 2

```
algoritmo()
{
        // declaracao das variaveis

        declare DEC: inteiro;       // um numero inteiro qulaquer de ate 5 digitos
        declare R1: inteiro;        // resto das divisoes de N por 2
        declare AUX: inteiro;       // uma variavel auxiliar cujo valor inicial e N
        declare BIN: cadeia;        // armazena a representacao binaria de N
        declare R2: cadeia;         // cadeia correspondente ao inteiro R1

        // inicializacao da condicao de contorno

        DEC := 1;       // caso contrario, nao entra no laco pois o valor inicial
                        // de qualquer variavel e zero.

        repita ateque ( DEC == 0 )
        {
                // leitura do numero decimal DEC

                leia ( "informe um numero inteiro: ", DEC );

                se ( DEC <> 0 )
                {
                        AUX := DEC;  // preservar o valor de N que deve ser mostrado

                        BIN := " ";     // inicializa a variavel B com espaco em branco

                        repita ateque ( AUX == 0 )
                        {
                                /* descricao do processo
                                        pega o resto da divisao de N por 2
                                        transforma em cadeia
                                        armazena na variavel B
                                        *****/
                                R1 := Resto ( AUX, 2 );
                                R2 := InteiroParaCadeia(R1);
                                BIN := concatena ( R2, BIN );
                                AUX := AUX / 2;             // proximo quociente
```

```
                    }
                    escreva ( "Base 10: ", DEC, " Base 2: ", BIN );
                }
        }
}
```

Problema 1.12.68b

Fazer um algoritmo que:
leia um conjunto de linhas contendo, cada uma, um número inteiro, na base 10, de até 5 dígitos. A última linha contem o valor zero. transforme esse número da base 10 para base 2 e escreva o número na base 10 e na base 2

```
algoritmo()
{
	// declaracao das variaveis

	declare DEC: inteiro;		// um numero inteiro qulaquer de ate 5 digitos
	declare R1: inteiro;		// resto das divisoes de N por 2
	declare AUX: inteiro;		// uma variavel auxiliar cujo valor inicial e N
	declare BIN: cadeia;		// armazena a representacao binaria de N
	declare R2: cadeia;		// cadeia correspondente ao inteiro R1

	// inicializacao da condicao de contorno

	DEC := 1;	// caso contrario, nao entra no laco pois o valor inicial
			// de qualquer variavel e zero.

	repita ateque ( DEC == 0 )
	{
		// leitura do numero decimal DEC

		leia ( "informe um numero inteiro: ", DEC );

		se ( DEC <> 0 )
		{
			AUX := DEC;
				// para preservar o valor de N que deve ser mostrado

			BIN := " ";	// inicializa a variavel B com espaco em branco

			repita ateque ( AUX == 0 )
			{
				R1 := Resto ( AUX, 2 );
					// pega o resto da divisao de N por 2
				R2 := InteiroParaCadeia(R1);
					// transforma em cadeia
				BIN := R2 + BIN;		// armazena na variavel B

				AUX := AUX / 2;		// proximo quociente
			}
			escreva ( "Base 10: ", DEC, " Base 2: ", BIN );
```

```
                }
        }
}
```

Problema 1.12.69

Fazer um algoritmo que:
- leia um conjunto de linhas contendo, cada uma, um número inteiro, na base 2. A última linha contem o valor zero.
- transforme esse número da base 2 para a base 10
- escreva o número na base 2 e na base 10

```
algoritmo()
{
        // declaracao das variaveis

        declare BIN: cadeia;        // armazena a cadeia binaria informada
        declare B2: inteiro;        // armazena o numerico na base 2
        declare DEC: inteiro;       // representacao decimal de BIN
        declare N: inteiro;         // o valor de cada um dos digitos da cadeia binaria
        declare AUX1: inteiro;      // comprimento da cadeia binaria
        declare AUX2: cadeia;       // cada um dos digitos da cadeia binaria BIN
        declare I: inteiro;         // o expoente da base

        // inicializacao da condicao de contorno

        B2 := 1;

        repita ateque ( B2 == 0 )
        {
                // leitura do numero na base 2

                leia ( "informe um numero na base 2: ", BIN );

                B2 := CadeiaParaInteiro(BIN);

                se ( B2 <> 0 )
                {
                        AUX1 := Comprimento (BIN);

                        // descobre quantos digitos tem na cadeia

                        DEC := 0;           // inicializa a variavel DEC com zero

                        I := 0;             // expoente inicial da base (2)

                        repita ateque ( AUX1 == 0 )
                        {
                                AUX2 := Subcadeia ( BIN, AUX1, 1 );
                                N := CadeiaParaInteiro ( AUX2 );
```

```
                        DEC := DEC + N * (2^I);
                        AUX1 := AUX1 - 1;
                        I := I + 1;
                    }
                    escreva ( "Base 2: ", BIN, " Base 10: ", DEC );
                }
            }
}
```

Manual de Referência do Interpretador

Hall é um interpretador de algoritmos.

Hall possui um conjunto de instruções pré programadas que permite aos seus usuários executarem seus programas. Para isso, ele possui um conjunto de palavras e instruções similares às encontradas nas linguagens de programação tradicionais.

Particularmente, sua sintaxe se assemelha à sintaxe da linguagem C, ou seja, o desenvolvimento dos algoritmos é baseado na divisão do programa em módulos, denominados funções, as quais são identificadas por parênteses ( ) colocados após o seu nome, e, suas instruções são agrupadas em blocos, os quais são delimitados por chaves { } e, cada comando do bloco é finalizado com um ponto e vírgula ;.

As linguagens mais modernas e poderosas tais como C++, Java, Delphi, Visual Basic, C++ Builder, etc, utilizam essa sintaxe ou variantes dela.

Sabemos da teoria de lógica de programação que um problema é resolvido através de um algoritmo, isto é, o espaço do problema é mapeado para o espaço do algoritmo fazendo-se abstrações em estruturas de dados e processos que manipulam essas estruturas de dados. As estruturas de dados são mapeadas em variáveis e os processos em funções.

Hall é estruturado, ou seja, ele permite a criação de funções isoladas contendo variáveis locais. Desse modo, os nossos algoritmos consistem em uma coleção de uma ou mais funções com variáveis locais. Uma função é composta de um nome, uma lista de parâmetros entre parênteses ( ) e o bloco de código associado. O bloco começa com um abre chaves { e, é seguido de uma ou mais instruções finalizadas com ponto e vírgula, e termina em um fecha chaves }. Uma instrução começa com uma palavra reservada, tais como: se, caso, repita,

enquanto, para ou então é uma expressão, tais como: x := 10, z := z + 2, leia(k), escreva (w).

Todos os nossos programas começam com uma chamada à função principal denominada algoritmo( ) e terminam com o último fecha chaves } dessa função. Quaisquer outras funções contidas no programa precisam ser chamadas direta ou indiretamente a partir da função principal. O fato de existir uma função principal é para informar ao interpretador Hall onde ele deve iniciar a execução do algoritmo.

Dentro da proposta inicial, a saber, a de definirmos um repertório de palavras, um subconjunto da língua Portuguesa, às quais seriam atribuida uma semântica especial para a descrição dos algoritmos, apresentamos abaixo as eleitas. Uma vez que essas palavras apresentam um significado previamente definido pelo interpretador, elas não podem ser utilizadas para outro fim, senão aquele a foi originalmente atribuido.

São elas, divididas em classes:

- *Palavras reservadas*
- *Funções de interface*
- *Funções matemáticas*
- *Funções de cadeia*
- *Funçoes de Arquivo*

Palavras reservadas

As palavras reservadas estão relacionadas com a definição de tipos e classes de dados e com as estruturas de controle do fluxo de execução, são elas:

tipo, caracter, cadeia, inteiro, real, Classe, básico, vetor, matriz, cubo, registro, conjunto, arquivo, fluxo, se, senao, selecao, caso, repita, ateque, enquanto, para, ate, passo, retorne, interrompa, continue

Símbolos reservados

Os símbolos reservados estão relacionadas com as operações matemáticas e outras operações definidas no interpretador

*+ - * / % ^ := = <- == # > >=*
< <= & | ! " ' . () {} [] , ;

Funções de interface: *As funções de interface são as instruções que nos permitem realizar um diálogo entre nosso programa e o usuário.*

- *Escreva( )*
- *Leia( )*
- *Posicao( )*
- *Moldura( )*
- *Pausa( )*
- *Tecla( )*
- *Direitos( )*
- *CopyRight( )*
- *LimpaTela ( );*
- *Imprima ( );*
- *Linha ( );*
- *Coluna ( );*
- *CorDoTexto ( );*
- *CorDoFundo ( );*

Sintaxe das funções de interface

1.Escreva( )

A função escreva tem por objetivo realizar a saida de informações para a tela.
A função escreva() não retorna valor. Ela é chamada a seguinte sintaxe:
Escreva (<expr>, ...);
Onde <expr> representa uma expressão qualquer, como por exemplo, uma cadeia ou uma variável. As reticências significam que a função aceita mais de um argumento e quando isso ocorrer, eles devem ser separados por vírgula.
Exemplos:
Escreva ("fernando");
Escreva ("o valor é: ", v);
Escreva ("o meu nome é ", x, "e o seu é ", y);

2.Leia( )

A função leia tem por objetivo realizar a entrada de informações para o processador. A função leia() não retorna valor.
Ela é chamada com a seguinte sintaxe:
Leia ([exprc], <var>);

Onde [exprc] é uma expressão do tipo cadeia e <var> é o nome de uma variável. Os colchetes [ ] envolvendo exprc indicam que essa expressão é opcional.
Exemplos:
Leia (x);
Leia ("informe o valor de b", b);

3.Posicao ( )

A função posicao() tem por objetivo posicionar o cursor na tela. A tela é dividida em 25 linhas e 80 colunas. A função posicao() não retorna valor.
Ela é chamada com a seguinte sintaxe:
Posicao (exprN1, exprN2); onde exprN1 é uma expressão numerica de valor inteiro, que representa a linha onde deverá ser posicionado o cursor e, exprN2 é uma expressão de valor inteiro, que representa a coluna onde deverá ser posicionado o cursor.
Exemplos:
Posicao (4, 40);
Posicao (L, C);

4.Moldura( )

A função moldura() tem por objetivo fazer um quadro com linhas duplas na tela. A função moldura() não retorna valor.
Ela é chamada com a seguinte sintaxe:
Moldura (exprN1, exprN2, exprN3, exprN4); onde exprN1, exprN2, exprN3 e exprN4 são expressões numéricas do tipo inteiro que representam as coordenadas dos cantos superior esquerdo e inferior direito do quadro. As coordenadas são especificadas em pares de números inteiros, onde o primeiro representa a linha e o segundo representa a coluna.
Exemplos:
Moldura (4, 5, 16, 60);
Moldura (L1, C1, L2, C2);

5.Pausa( )

A função pausa() tem por objetivo fazer uma interrupção na execução do algoritmo e aguardar a tecla <enter> ser pressionada. A função pausa() não retorna valor.

Ela é chamada com a seguinte sintaxe:
Pausa(); A função não tem argumentos.
Exemplos:
Pausa();
Escreva ("tecle <enter> para continuar"); Pausa();

6.Tecla( )

A função tecla() tem por objetivo fazer uma interrupção na execução do algoritmo e aguardar até que uma tecla qualquer seja pressionada. Quando então, uma tecla é pressionada, a função tecla() retorna a tecla que foi pressionada. O valor de retorno da função é do tipo caracter.
Exemplos:
C := tecla( ); onde C deve ser uma variável do tipo caracter.

7.Direitos( )

A função direitos() tem por objetivo exibir a tela de direitos autorais do interpretador. A função direitos() não retorna valor.
Ela é chamada com a seguinte sintaxe:
Direitos();

8.CopyRight( )

A função CopyRight() tem por objetivo exibir a mensagem de copyright do interpretador. A função copyright() não retorna valor.
Ela é chamada com a seguinte sintaxe:
CopyRight();

9.LimpaTela ( );

A função LimpaTela() tem por objetivo limpar a tela do vídeo. A função não retorna valor.
Ela é chamada com a seguinte sintaxe:
LimpaTela();

10.Imprima ( );

A função Imprima() tem por objetivo enviar a saida de informações para a impressora. A função imprima() não retorna valor.
Ela é chamada com a seguinte sintaxe:
Imprima (Expr, ...); onde ExprC é uma expressão do tipo cadeia e Var é o nome de uma variável.
Exemplos:
Imprima ("Testando a impressora");
Imprima ("O preco do produto e ", P);

11.Linha ( );

A função linha() tem por objetivo retorna o valor da linha na qual encontra-se posicionado o cursor. A função linha() retorna um valor do tipo inteiro entre 1 e 25.
Ela é chamada com a seguinte sintaxe:
N := Linha( ); onde N deve ser uma variável do tipo inteiro.
Exemplos:
L := Linha( );
Escreva ("A linha onde esta o cursor e ", L);

12.Coluna ( );

A função coluna() tem por objetivo retorna o valor da coluna na qual encontra-se posicionado o cursor. A função coluna() retorna um valor do tipo inteiro entre 1 e 80.
Ela é chamada com a seguinte sintaxe:
N := Coluna( ); onde N deve ser uma variável do tipo inteiro.
Exemplos:
C := Coluna( );
Escreva ("A coluna onde esta o cursor e ", C);

13.CorDoTexto ( );

A função CorDotexto() tem por objetivo estabelecer a cor com que será exibido os textos com a função escreva(). A função CorDoTexto() não retorna

valor. A função espera um argumento que representa uma cor a ser estabelecida de um universo de 15 cores.
A função é chamada com a seguinte sintaxe:
CorDoTexto (10);
CorDoTexto (C); onde C representa o valor de uma cor

14.CorDoFundo ( );

A função CorDoFundo() tem por objetivo estabelecer a cor de fundo com que será exibido os textos com a função escreva(). A função CorDoFundo() não retorna valor. A função espera um argumento que representa uma cor a ser estabelecida de um universo de 15 cores.
A função é chamada com a seguinte sintaxe:
CorDoFundo (10);
CorDoFundo (C); onde C representa o valor de uma cor

Funções matemáticas: *As funções matemáticas são as instruções que nos permitem utilizar de recursos matemáticos freqüentemente encontrados e solicitados na resolução de algum problema. Elas foram implementadas porque dessa forma tornam mais fácil realizar os cálculos, o que de outra forma deveria ser desenvolvido pelo usuário.*

- *Sinal( )*
- *ValorAbs( )*
- *ParteInteira( )*
- *ParteDecimal( )*
- *Quociente( )*
- *Resto( )*
- *Raiz( )*
- *Potencia( )*
- *Seno( )*
- *Coseno( )*
- *Tangente( )*
- *ArcoSeno ( )*
- *ArcoCoseno ( )*
- *ArcoTangente ( )*
- *Exponencial( )*
- *LogDec( )*
- *LogNeper( )*
- *Logaritmo ( )*

1.Sinal( )

A função sinal() retorna o valor inteiro 1 se o argumento passado à função for positivo e, retorna –1 se o argumento passado à função for negativo.
A função é chamada com a seguinte sintaxe:
S := Sinal (exprN); onde exprN é uma expressão de classe numérica (tipo inteiro ou real) e S é uma variável do tipo inteiro.

2.ValorAbs( )

A função ValorAbs() retorna o valor absoluto de seu argumento, isto é, o seu módulo. O argumento deve ser uma expressão de classe numérica (do tipo inteiro ou real).
Ela é chamada com a seguinte sintaxe:

V := ValorAbs (exprN); onde exprN é uma expressão numérica, inteiro ou real, e V é uma variável numérica. A variável V deve ser do mesmo tipo de exprN para que o valor de retorno não seja alterado.
Exemplos:
N := ValorAbs (-5);

3.ParteInteira( )

A função ParteInteira() retorna a parte inteira de seu argumento, isto é, os dígitos decimais são ignorados.
Ela é chamada com a seguinte sintaxe:
N := ParteInteira (exprN);

4.ParteDecimal( )

A função ParteDecimal() retorna a parte decimal de seu argumento, isto é, os dígitos inteiros são ignorados.
Ela é chamada com a seguinte sintaxe:
N := ParteDecimal (exprN);

5.Quociente( )

A função Quociente() retorna o quociente entre seus argumentos. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
Q := Quociente (exprN1, exprN2); onde exprN1 e exprN2 são expressões numéricas e exprN2 deve ser diferente de zero. A função retorna o valor de exprN1 / exprN2.
Exemplos:
Q := Quociente (x, y);

6.Resto( )

A função Resto() retorna o resto da divisão inteira entre seus argumentos. O valor de retorno é do tipo inteiro.
Ela é chamada com a seguinte sintaxe:

R := Resto (exprN1, exprN2); onde exprN1 e exprN2 são expressões numéricas e exprN2 deve ser diferente de zero. A função retorna o valor do resto de exprN1 / exprN2.
Exemplos:
R := Resto (x, y);

7.Raiz( )

A função Raiz() retorna a raiz quadrada de seu argumento. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
R := Raiz (exprN); onde exprN é expressão numérica e exprN deve ser positivo.
Exemplos:
R := Raiz (x);

8.Potencia( )

A função Potencia() retorna a potenciação entre seus argumentos. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
P := Potencia (exprN1, exprN2); onde exprN1 e exprN2 são expressões numéricas. A função retorna o valor de exprN1 elevado a exprN2.
Exemplos:
P := Potencia (2, 5);

9.Seno( )

A função Seno() retorna o seno de seu seu argumento. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
S := Seno (exprN, T); onde exprN é o valor do arco ou ângulo e o segundo argumento T especifica de o valor de exprN que está sendo passado está em graus ou radianos.
Exemplos:
S := Seno (a, "G");
S := Seno (a, "R");

10.Coseno( )

A função CoSeno() retorna o cosseno de seu seu argumento. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
C := CoSeno (exprN, T); onde exprN é o valor do arco ou ângulo e o segundo argumento T especifica de o valor de exprN que está sendo passado está em graus ou radianos.
Exemplos:
S := CoSeno (a, "G");
S := CoSeno (a, "R");

11.Tangente( )

A função Tangente() retorna a tangente de seu seu argumento. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
T := Tangente (exprN, T); onde exprN é o valor do arco ou ângulo e o segundo argumento T especifica de o valor de exprN que está sendo passado está em graus ou radianos.
Exemplos:
T := Tangente (a, "G");
S := Tangente (a, "R");

12.ArcoSeno ( )

A função ArcoSeno() retorna o arco ou ângulo cujo seno é o seu argumento. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
A := ArcoSeno (exprN, T); onde exprN é o valor do seno do arco ou ângulo que se deseja saber, e o segundo argumento T especifica se o valor de retorno da função deverá ser dado em graus ou radianos.
Exemplos:
A := ArcoSeno (a, "G");
A := ArcoSeno (a, "R");

13.ArcoCoseno ( )

A função ArcoCoSeno() retorna o arco ou ângulo cujo cosseno é o seu argumento. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
A := ArcoCoseno (exprN, T); onde exprN é o valor do cosseno do arco ou ângulo que se deseja saber, e o segundo argumento T especifica se o valor de retorno da função deverá ser dado em graus ou radianos.
Exemplos:
A := ArcoCoSeno (a, "G");
A := ArcoCoSeno (a, "R");

14.ArcoTangente ( )

A função ArcoTangente() retorna o arco ou ângulo cuja tangente é o seu argumento. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
A := ArcoTangente (exprN, T); onde exprN é o valor da tangente do arco ou ângulo que se deseja saber, e o segundo argumento T especifica se o valor de retorno da função deverá ser dado em graus ou radianos.
Exemplos:
A := ArcoTangente (a, "G");
A := ArcoTangente (a, "R");

15.Exponencial ( )

A função Exponencial() retorna uma potência do número de nepper cujo expoente é o seu argumento. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
E := Exponencial (exprN); onde exprN é uma expressão numérica a qual se deseja elevar o número e.
Exemplos:
E := Exponencial (3);

16.LogDec ( )

A função LogDec() retorna o logaritmo decimal da expressão que foi passada como argumento para a função. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:

L := LogDec (exprN); onde exprN é uma expressão numérica a qual se deseja o logarítmo decimal.
Exemplos:
L := LogDec (123);

17.LogNeper ( )

A função LogNeper() retorna o logaritmo neperiano, da expressão que foi passada como argumento para a função. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
L := LogNeper (exprN); onde exprN é uma expressão numérica a qual se deseja o logarítmo neperiano.
Exemplos:
L := LogNeper (2);

18.Logaritmo ( )

A função Logaritmo() retorna o logaritmo, em uma base qualquer,decimal, neperiano ou outra, da expressão que foi passada como argumento para a função. O valor de retorno é do tipo real.
Ela é chamada com a seguinte sintaxe:
L := Logaritmo (exprN1, exprN2); onde exprN1 é uma expressão numérica a qual se deseja o logarítmo, e exprN2 é uma expressao numerica, do tipo inteiro ou rela, que representa a base.
Exemplos:
L := Logaritmo (8, 10);
L := Logaritmo (8, e);

Funções de cadeia: *As funções de cadeia são as instruções que nos permitem tratar seqüência de caracteres. As cadeias são recursos bastante utilizados na programação em geral.*

- *Comprimento( )*
- *Concatena( )*
- *SubCadeia( )*
- *Primeiros( )*
- *Ultimos( )*
- *Compara( )*
- *Maiusculas( )*
- *Minusculas( )*
- *Insere( )*
- *SobrePoe( )*
- *CadeiaParaInteiro( )*
- *CadeiaParaReal( )*
- *InteiroParaCadeia( )*
- *RealParaCadeia( )*

1.Comprimento( )

A função comprimento() retorna o comprimento do argumento em quantidade de caracteres. O valor retornado pela função é do tipo inteiro.
Ela é chamada com a seguinte sintaxe:
C := Comprimento (exprC); onde exprC é uma expressão do tipo cadeia e C representa uma variável qualquer, de classe numerica, que receberá o valor retornado pela função.
Exemplos:
C := Comprimento ("Dominando a linguagem C");

2.Concatena( )

A função concatena() retorna os seus argumentos concatenados em uma única cadeia, sem espaços entre eles. O valor retornado pela função é do tipo cadeia.
Ela é chamada com a seguinte sintaxe:
A := Concatena (exprC, ...); onde exprC é uma expressão do tipo cadeia e A representa uma variável qualquer, do tipo cadeia, que receberá o valor retornado pela função. As reticências significam que a função pode receber vários argumentos, sendo eles separados por vírgulas.

Exemplos:
A := Concatena ("Quem tem boca", " ", "vai a Roma");

3.SubCadeia( )

A função SubCadeia() retorna uma subcadeia de seu argumento. O valor retornado pela função é do tipo cadeia.
Ela é chamada com a seguinte sintaxe:
S := SubCadeia (exprC, exprN1, exprN2); onde exprC é uma expressão do tipo cadeia da qual se deseja uma subcadeia, exprN1 é uma expressão do tipo inteiro que representa a posição do primeiro caracter da subcadeia e, exprN2 representa a quantidade de caracteres dessa subcadeia.
Exemplos:
S := SubCadeia ("Tecnicas Estruturadas", 5, 4);

4.Primeiros( )

A função Primeiros() retorna uma subcadeia de seu argumento. O valor retornado pela função é do tipo cadeia.
Ela é chamada com a seguinte sintaxe:
S := Primeiros (exprC, exprN); onde exprC é uma expressão do tipo cadeia da qual se deseja uma subcadeia, exprN é uma expressão do tipo inteiro que representa a quantidade de caracteres de exprC, contados a partir de seu início, que se deseja retornar.
Exemplos:
S := Primeiros ("Tecnicas Estruturadas", 4);

5.Ultimos( )

A função Ultimos() retorna uma subcadeia de seu argumento. O valor retornado pela função é do tipo cadeia.
Ela é chamada com a seguinte sintaxe:
S := Ultimos (exprC, exprN); onde exprC é uma expressão do tipo cadeia da qual se deseja uma subcadeia, exprN é uma expressão do tipo inteiro que representa a quantidade de caracteres de exprC, contados a partir de seu final, que se deseja retornar.
Exemplos:
S := Ultimos ("Tecnicas Estruturadas", 4);

6.Compara( )

A função Compara() tem por objetivo comparar duas cadeia com relação a ordem alfabética de seus argumentos. Ela retorna um valor inteiro, negativo, nulo ou positivo de acordo com a ordem de passagem de seus argumentos.
Ela é chamada com a seguinte sintaxe:
S := Compara (exprC1, exprC2); onde exprC1 e exprC2 são expressões do tipo cadeia. Se os valores representados pelas expressões exprC1 e exprC2 forem iguais o valor de retorno é 0 (zero), se exprC2 for maior que exprC1, isto é, exprC2 representar um valor que está em uma posição à frente de exprC1, considerando a ordem alfabética, o valor de retorno será 1 (um), caso contrário o valor de retorno será –1. Dito de outra forma, o valor de retorno será negativo se a ordem dos argumentos estiver diferente da ordem alfabética dos mesmos.
Exemplos:
P := Compara ("Fernando", "Rosa");

7.Maiusculas( )

A função Maiusculas() tem por objetivo retornar o seu argumento transformado em maiusculas. O valor de retorno é do tipo cadeia.
Ela é chamada com a seguinte sintaxe:
S := Maiusculas (exprC); onde exprC é uma expressão do tipo cadeia.
Exemplos:
S := Maiusculas ("amanda");

8.Minusculas( )

A função Minusculas() tem por objetivo retornar o seu argumento transformado em minusculas. O valor de retorno é do tipo cadeia.
Ela é chamada com a seguinte sintaxe:
S := Minusculas (exprC); onde exprC é uma expressão do tipo cadeia.
Exemplos:
S := Minusculas ("marina");

9.Insere( )

10.SobrePoe( )

11.CadeiaParaInteiro( )

A função cadeiaParaInteiro() tem por objetivo retornar um numérico do tipo inteiro correspondente a cadeia numérica de seu argumento.
Ela é chamada com a seguinte sintaxe:
V := CadeiaParaInteiro (exprC); onde exprC é uma expressão do tipo cadeia.
Se a cadeia especificada em exprC não puder ser convertida em inteiro, o valor de retorno será 0 (zero) e V representa uma variável do tipo inteiro.
Exemplos:
V := CadeiaParaInteiro ("125");

12.CadeiaParaReal( )

A função cadeiaParaReal() tem por objetivo retornar um numérico do tipo real correspondente a cadeia numérica de seu argumento.
Ela é chamada com a seguinte sintaxe:
V := CadeiaParaReal (exprC); onde exprC é uma expressão do tipo cadeia e, V representa uma variável do tipo real.
Se a cadeia especificada em exprC não puder ser convertida em real, o valor de retorno será 0 (zero)
Exemplos:
V := CadeiaParaReal ("3.141592");

13.InteiroParaCadeia( )

A função InteiroParaCadeia() tem por objetivo retornar uma cadeia numérica do tipo inteiro correspondente ao inteiro que foi passado como seu argumento.
Ela é chamada com a seguinte sintaxe:
S := InteiroParaCadeia (exprN); onde exprN é uma expressão do tipo inteiro e, S representa uma variável do tipo cadeia.
Se a cadeia especificada em exprN não puder ser convertida em inteiro, o valor de retorno será espaços.
Exemplos:
V := InteiroParaCadeia (125);

14.RealParaCadeia( )

A função RealParaCadeia() tem por objetivo retornar uma cadeia numérica do tipo real correspondente ao real que foi passado como seu argumento.
Ela é chamada com a seguinte sintaxe:
S := RealParaCadeia (exprN); onde exprN é uma expressão do tipo real e, S representa uma variável do tipo cadeia.
Se a cadeia especificada em exprN não puder ser convertida em real, o valor de retorno será espaços.
Exemplos:
V := RealParaCadeia (3.141592);

Funções de arquivos: *As funções para o tratamento de arquivo são as instruções que nos permitem tratar informações que poderão ser armazenadas indefinidamente no computador, isto é, elas tratam da manipulação da informação que está armazenada no winchester por exemplo.*

- *Vincular ( );*
- *Associar ( );*
- *Criar ( );*
- *Abrir ( );*
- *Fechar ( );*
- *Gravar ( );*
- *Regravar ( );*
- *Deletar ( );*
- *Ler ( );*
- *Posicionar ( );*
- *FDA ( );*

1.Vincular ( );

A função Vincular() tem por objetivo estabelecer uma ligação entre um identificador lógico que será utilizado no texto do algoritmo e o correspondente identificador físico do arquivo em disco.
Ela é chamada com a seguinte sintaxe:
Vincular (exprC1, exprC2); onde exprC1 representa o nome lógico do arquivo utilizado no algoritmo e, exprC2 representa o nome físico do arquivo no disco.
Exemplos:
Vincular (clientes, "arq001.dat");

2.Associar ( );

A função Associar() tem por objetivo estabelecer uma ligação entre um identificador lógico que será utilizado no texto do algoritmo e o correspondente registro que representa as informações armazenadas.
Ela é chamada com a seguinte sintaxe:
Associar (exprC1, exprC2); onde exprC1 representa o nome lógico do arquivo utilizado no algoritmo e, exprC2 representa o nome do registro que representa as informações armazenadas.
Exemplos:
Associar (clientes, pessoas);

3.Criar ( );

A função Criar() tem por objetivo criar o arquivo físico no disco.
Ela é chamada com a seguinte sintaxe:
Criar (exprC); onde exprC representa o identificador lógico do arquivo ou o seu identificador físico.
Exemplos:
Criar (clientes);
Criar ("arq001.dat");

4.Abrir ( );

A função Abrir() tem por objetivo abrir o arquivo físico no disco.
Ela é chamada com a seguinte sintaxe:
Abrir (exprC); onde exprC representa o identificador lógico do arquivo ou o seu identificador físico.
Exemplos:
Abrir (clientes);
Abrir ("arq001.dat");

5.Fechar ( );

A função Fechar() tem por objetivo fechar o arquivo físico no disco.
Ela é chamada com a seguinte sintaxe:
Fechar (exprC); onde exprC representa o identificador lógico do arquivo ou o seu identificador físico.
Exemplos:
Fechar (clientes);
Fechar ("arq001.dat");

6.Gravar ( );

A função Gravar() tem por objetivo gravar as informações no arquivo físico.
Ela é chamada com a seguinte sintaxe:
Gravar (exprC1, exprC2); onde exprC1 representa o identificador lógico ou físico do arquivo e exprC2 representa a informação que se deseja armazenar.
Exemplos:

Gravar (clientes, "teste de gravacao");
Gravar ("arq001.dat", "teste de gravacao");

7.Regravar ( );

A função ReGravar() tem por objetivo regravar as informações no arquivo físico.
Ela é chamada com a seguinte sintaxe:
ReGravar (exprC1, exprC2); onde exprC1 representa o identificador lógico ou físico do arquivo e exprC2 representa a informação que se deseja armazenar.
Exemplos:
ReGravar (clientes, "novo teste de gravacao");
ReGravar ("arq001.dat", "novo teste de gravacao");

8.Deletar ( );

A função Deletar() tem por objetivo excluir informações no arquivo físico.
Ela é chamada com a seguinte sintaxe:
Deletar (exprC1, exprC2); onde exprC1 representa o identificador lógico ou físico do arquivo e exprC2 representa a informação que se deseja deletar.
Exemplos:
Deletar (clientes, pessoas);
Deletar ("arq001.dat", pessoas);

9.Ler ( );

A função Ler() tem por objetivo recuperar informações no arquivo físico.
Ela é chamada com a seguinte sintaxe:
Ler (exprC1, exprC2); onde exprC1 representa o identificador lógico ou físico do arquivo e exprC2 representa a informação que se deseja ler.
Exemplos:
Ler (clientes, pessoas);
Ler ("arq001.dat", pessoas);

10.Posicionar ( );

A função Posicionar() tem por objetivo permitir o posicionamento em uma determinada informação no arquivo físico.
Ela é chamada com a seguinte sintaxe:
Posicionar (exprC, exprN); onde exprC representa o identificador lógico ou físico do arquivo e exprN a posição em que se deseja apontar no arquivo.
Exemplos:
Posicionar (clientes, INICIO);
Posicionar ("arq001.dat", FINAL);

11.FDA ( );

A função fda() retorna um valor indicando se o arquivo está posicionado em seu final.
Ela é chamada com a seguinte sintaxe:
FDA (exprC); onde exprC representa o identificador lógico ou físico do arquivo
Exemplos:
FDA (clientes);
FDA ("arq001.dat");

Referência Bibliográfica

- Algoritmos Estruturados

Harry Farrer, Christiano Gonçalves Becker, Eduardo Chaves Faria, Helton Fábio de Matos, Marcos Augusto dos Santos, Miriam Lourenço Maia da Editora Guanabara, 2ª edição.