Elidian Alencar -- Compiladores -- Flex-Bison -- Codigo Calc

A linguagem **INCONI** foi desenvolvida na disciplina de compiladores.

MARCADORES DE INÍCIO E FIM DO PROGRAMA

O comando ***in*** inicia o programa, já o comando ***ni*** o finaliza

```
in

ni
```

COMENTÁRIO

Para comentar uma linha, adicione ***#*** no inicio do comentário

```
in
    # código
    # ...
    # código
ni
```

Para comentar várias linhas adicione ***/**** no início e ****/*** no final do comentário.

```
in
    /*
    código
    ...
    código
    */
ni
```

TIPOS E DECLARAÇÕES DE VARIAVEIS

As declarações devem ser precedidas de tipo e seguidas pelo nome da variável, podendo ser atribuido valor durante a declaração.

real: aceita número real

```
real val
real val2 = 10.0
real val_3 = val2, val4
```

inteiro: aceita número natural

```
inteiro a
inteiro b = 5
inteiro c = b, d
```

texto: aceita um texto/string

```
texto name
texto name2 = "nome"
texto my_name = name2, sobrenome
```

obs: 1. O nome das variaveis devem iniciar sempre com uma letra e depois podem conter letra, numero e/ou underline (_). 2. A declaração de um mesmo tipo pode ser em linha separados por vírgula (,).

DECLARAÇÕES DE VETORES

Semelhante as declarações anteriores

```
real vec[3]
real vec2[5] = {2, 3, 5, 7, 11}

inteiro vec_3[2] = {3, 6}, vec4[4]
inteiro vec5[3] = vec2

texto name[1]
texto name2[3] = {"", "nome"}
texto my_name[2] = {name2[1], "sobrenome"}
```

ATRIBUIÇÃO DE VALORES

Para atribuição utiliza-se o comando *=* após a variável

```
int a, soma
real b, c[2]
texto nome[2], primeiro

a = 5
b = 10.7
c[0] = a + b
soma = c[0]
nome[0] = "nome"
nome[1] = "sobrenome"
primeiro = nome[0]
```

OPERAÇÃO DE ESCRITA NA TELA

Para escrita em tela utiliza-se o comando *>>*

```
inteiro a = 1
real b = 5.5
texto name[1] = "primeiro segundo"

>> "Hello World!"
>> a
>> 5.0
>> a + b
>> name[0]
```

Saída

```
Hello World!
1
5.00
6.50000
primeiro segundo
```

A escrita pode ser em uma mesma linha separados por vírgula (,)

```
>> "Hello World!", ' ', a, ' ', 5.0 , ' ', a + b, ' ', name[1]
```

Saída

```
Hello World! 1 5.00 6.50000 primeiro segundo
```

Obs.: Por padrão, ao final do escrever há um salto de linha.

Há duas formas para pular linha:

1. escrevendo um texto com apenas *"\n"*;
2. escrevendo *>>;*.

Exemplo:

```
>> "Hello", "\n", "World!"
>>;
>> "fim"
```

Saída:

```
Hello
World!

fim
```

OPERAÇÃO DE LEITURA

Para receber informação da tela utiliza-se o comando **<<**

```
real idade
>> "Digite sua idade: "

idade <<
```

OPERAÇÕES MATEMÁTICAS

```
real a = 5.0
inteiro b = 10

>> "Soma: ", 5.0 + b
>> "Subtracao: ", a - 10
>> "Multiplicacao: ", 5.0 * 10
>> "Divisao: ", a / b
>> "Resto de divisao: ", 5 % 2
>> "Exponenciacao: ", 5.0 ^ 2
>> "Radiciacao: ", raiz(4)
>> "Seno: ", sin(5)
>> "Cosseno: ", cos(5)
>> "Modulo: ", |-15|
```

Saída:

```
Soma: 15.00000
Subtracao: -5.00000
Multiplicacao: 50.00000
Divisao: 0.50000
Resto de divisao: 1.00000
Exponenciacao: 25.00000
Radiciacao: 2.00000
Seno: 0,08715
Cosseno: 0,99619
Modulo: -15.00000
```

OPERAÇÕES LÓGICAS

```
inteiro a = 10, b = 5, c = 15

>> "Maior: ", a > b
>> "Maior igual: ", b >= c
>> "Menor: ", c < a
>> "Menor igual: ", a <= c
>> "Igual: ", c == b
>> "Diferente: ", b != a
>> "OU: ", a > b !! a == c
>> "E: ", b > a && b < c
```

Saída:

```
 Maior: 1
Maior igual: 0
Menor: 0
Menor igual: 1
Igual: 0
Diferente: 1
OU: 1
E: 1
```

ORDEM DE PRECEDÊNCIAS

1. Parênteses
2. Exponenciação
3. Multiplicação e divisão tem mesmo grau de precedência
4. Soma e subtração tem mesmo grau de precedência
5. Expressões lógicas

Em uma equação simples:

```
>> a / b + a * b - 1
```

A precedência é feita como segue abaixo:

```
>> (((a / b) + (a * b)) - 1)
```

Passo a passo:

```
>> (a/b) # divisão
>> (a*b) # produto
>> (a/b + a*b) # soma
>> ((a/b + a*b) - 1) # subtração
```

Exemplo de lógica:

```
inteiro a = 5
>> a * 2 < 5
```

Saída:

```
0
```

EXTRUTURAS DE DECISÃO *IF*, *IF ELSE* E *OPERADOR TERNÁRIO*

IF

Estrutura:

```
inif codigo_logico
{
  # lista de codigos caso IF POSITIVO
}
```

Exemplo:

```
real a = 3
inteiro b = 7
inif 2*a > b {
  >> "SIM"
}
```

Saída:

```

```

IF ELSE

Estrutura:

```
inif codigo_logico
{
  # lista de codigos caso IF POSITIVO
}
infi
{
  # lista de codigos caso IF NEGATIVO
}
```

Exemplo:

```
inteiro a = 3
real b = 7
inif a*2 > b
{
  >> "SIM"
}
infi
{
  >> "NAO"
}
```

Saída:

```
NAO
```

OPERADOR TERNÁRIO

Estrutura:

```
codigo_logico ? codigo_caso_positivo : codigo_caso_negativo
```

Exemplo:

```
5 >= 10 ? >> "codigo caso positivo" : >> "codigo caso negativo"
```

Saída:

```
codigo caso negativo
```

ESTRUTURAS DE REPETIÇÃO *WHILE* E *FOR*

WHILE

Estrutura:

```
inwhile codigo_logico
{
  # lista de codigos caso teste WHILE POSITIVO
}
```

Exemplo:

```
inteiro x = 0
inwhile x<=10-x
{
  >> x
  x = x + 1
}
```

Saída:

```
0
1
2
3
4
5
```

FOR

Estrutura:

```
infor codigo_atribuição : codigo_logico : codigo_aritmetico
{
  # lista de codigos caso teste FOR POSITIVO
}
```

Exemplo:

```
infor inteiro x = 2 : x <= 10 - x : 1
{
  >> x
}
```

Saída:

```
2
3
4
5
```

FUNÇÕES

Estrutura:

```
tipo nome_da_funcao (){
  # codigo
  inout codigo_logico_aritmetico_valor
}

tipo nome_da_funcao (tipo parametro){
  # codigo
  inout codigo_logico_aritmetico_valor
}

tipo nome_da_funcao (tipo parametro_1, tipo parametro_2){
  # codigo
  inout codigo_logico_aritmetico_valor
}
```

Tipos de funções:

1. inteiro
2. real
3. void

Obs.: Não é obrigatório o uso da palavra chave de retorno ***inout***. Assim, têm-se alguns casos:

1. a expressão ***inout*** é ignorada na função do tipo void, retornando 0 (zero) ao final;
2. caso seja usado ***inout*** a função não void será interrompida e retornará o valor após ela;
3. caso não seja usado ***inout*** a função continuará até seu término e ao final retornará 0 (zero).

Exemplo:

```
void soma (){
  >> 10 + 15
}

inteiro dobro (real n){
  inout 2 * n
}

real produto (inteiro a, real b){
  real res = a * b
  inout res
}

soma()
>> dobro(3)
>> produto( dobro(3), 5)
```

Saída:

```
25.000000
6.000000
30.000000
```

EXEMPLO 1

```
in

  real b, h, area

  >> "Calculo da area do retangulo"
  >> "" # pular linha

  >> "Digite a base: "
  b <<

  >> "Digite a altura: "
  h <<

  >>;
  area = h * b
  >> "A area do retangulo eh: ", area

ni
```

EXEMPLO 2

```
in

    real juroscomposto(real c, real i, real t){
        i =  i/100
        #>> "M ", c*((1+(i))^t)
        inout c*(1+i)^t
    }

    real prestacao_price(real c, real i, real t){
        i = i/100
        #>> "P ", (c*((1+i)^t)*i)/((1+i)^t-1)
        inout c*(1+i)^t*i/((1+i)^t-1)
    }
```

```
}

real capital, montante, juros, taxa
inteiro periodo, op
texto nome

>> "SIMULADOR DE EMPRESTIMOS"
>> "Qual seu nome?"
nome <<
>> "Qual o valor de Capital desejado?"
capital <<
>> "Qual a duracao do pagamento em meses?"
periodo <<
>> "Qual a taxa de juros ao mes?"
taxa <<
>>;

montante = juroscomposto(capital, taxa, periodo)

>> "Vejamos, Sr(a) ", nome, ","
>> "-> INVESTIMENTO <-"
>> "O Capital aplicado renderia um Montante de ", montante, " reais se no regime do juros composto."
>> "Sendo o Montante igual a soma do Capital e do Juros composto total."
>> "\n", "----------------------"
>> "Resumo"
>> "----------------------"
>> "Capital  = ", capital
juros = montante - capital
>> "Juros    = ", juros
>> "----------------------"
>> "Montante = ", montante
>> "----------------------"

>>;
>> "-> EMPRESTIMO <-"
>> "Qual o sistema de amortizacao?"
>> "(1) Tabela PRICE"
>> "(2) Tabela SAC"
>> "(3) Sair"
op <<


inteiro t
real prestacao1
real j[periodo + 1], a1[periodo + 1], d[periodo + 1]
real prestacao2[periodo + 1]
real a2
real total[4]

inwhile(op==1 || op ==2){
    total[0] = 0
    total[1] = 0
    total[2] = 0
    total[3] = capital
    inif op == 1 {
        prestacao1 = prestacao_price(capital, taxa, periodo)
        t = 0
        d[0] = capital

        >>;
        >> "A prestacao no modelo PRICE eh ", prestacao1
        >>;
        >> "TABELA PRICE"
        >> "-------------------------------------------------------------------"
        >> "Mes   | Prestacao    | Juros    | Amortizacao    | Saldo Devedor "
        >> "-------------------------------------------------------------------"
```

```
            >> "0     |              |          |               | ", d[0]
            inwhile t < periodo {
                t++
                j[t] = d[t - 1]*taxa/100
                a1[t] = prestacao1 - j[t]
                d[t] = d[t - 1] - a1[t]
                total[0] = total[0] + prestacao1
                total[1] = total[1] + j[t]
                total[2] = total[2] + a1[t]
                total[3] = d[t]
                >> t, "     | ", prestacao1 ,"       | ", j[t] ,"   | ", a1[t] ,"       | ", d[t]
            }
        } infi {
            inif op == 2 {
                a2 = capital/periodo
                t = 0
                d[0] = capital
                prestacao2 = a2 + d[0]*taxa/100

                >>;
                >> "A prestacao no modelo SAC comeca em ", prestacao2
                >>;
                >> "TABELA PRICE"
                >> "-------------------------------------------------------------------------"
                >> "Mes   | Prestacao    | Juros    | Amortizacao | Saldo Devedor "
                >> "-------------------------------------------------------------------------"
                >> "0     |              |          |             | ", d[0]
                inwhile t < periodo {
                    t++
                    j[t] = d[t - 1]*taxa/100
                    prestacao2[t] = a2 + j[t]
                    d[t] = d[t - 1] - a2
                    total[0] = total[0] + prestacao2[t]
                    total[1] = total[1] + j[t]
                    total[2] = total[2] + a2
                    total[3] = d[t]
                    >> t, "     | ", prestacao2[t] ,"       | ", j[t] ,"   | ", a2 ,"       | ", d[t]
                }
            }
        }
        >> "-------------------------------------------------------------------------"
        >> "Total     | ", total[0] ,"       | ", total[1] ,"   | ", total[2] ,"       | ", total[3]

        >>;
        >> "-> EMPRESTIMO <-"
        >> "Qual o sistema de amortizacao?"
        >> "(1) Tabela PRICE"
        >> "(2) Tabela SAC"
        >> "(3) Sair"
        op <<
    }
ni
```