Declarações

```
<tipo da var de entrada>  *pmar;     // declara pmar como apontador
                                     // como em float *pmar;
```

Expressões

```
&sat   significa "endereço da variável sat"
          como em pmar=&sat;

*pmar significa "conteúdo da var apontada por pmar"
          como em *pmar+11.55;
```

FUNÇÕES

```
   <tipo da var de saída> nome_func ( <tipo da var de entrada> <nome_var>, ... ){
                  <corpo da função>
                  }
```

(1) parâmetro x, SEM *, entre parênteses na declaração de função F(<tipo> x),
EXIGE x na chamada de F(), SEM &.
Uma cópia do valor de x é feita dentro do corpo de F(), cópia LOCAL à função.
Na chamada de F(), ao invés de x pode ocorrer uma expressão cujo valor é
calculado ANTES de ser copiado dentro do corpo de F();

```
int Lua( float x, int t){
 .............
 ....... x+3.81.........
}
main(){
float h, bk;  int t2;
.............................
h=Lua(3.2+bk, t2);
.....................................
```

(2) parâmetro *x, com * entre parênteses na declaração de função F(<tipo> *x),
EXIGE &x na chamada de F(), COM &. NÃO pode ser uma expressão
O endereço &x é "passado" para a função F().

```
int Lua( float *x, int t){
 .............
 ....... *x+3.81.........
}
main(){
float h, bk;  int t2;
.............................
h=Lua( &bk, t2);
.....................................
```

(3) void F(...) na declaração da função F() EXIGE que return NÃO ocorra no
corpo de F()

(4) char F(...) na declaração da função F() EXIGE que ocorra pelo menos um return x;
no corpo de F(), onde x seja uma variável do tipo char. Analogamente para outros
tipos de variável.

```
char Lua( ... ){
 .............
 if( ... ) return 'n';
}
main(){
char c2;
.............................
c2= Lua( ... );
.................................
```