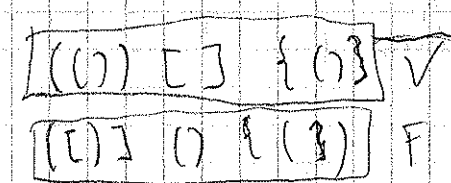
Claúdia Gomes, nº 57803 30/04/2010

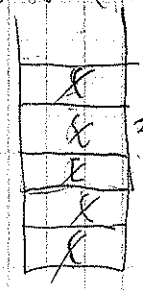
Listas de Alunos

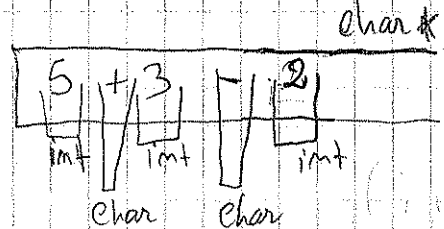
```
typedef struct sAluno {
   char nome[60];
   char numero[10];
} Aluno;                          } Comprimento fixo.

typedef struct sLAluno {
   int nalunos;
   Aluno lista[100];
} LAluno;

int main(){
Aluno a1={...}, a2={...}, a3={...};
LAluno l1 = {3, {a1, a2, a3}}, l2;
   ...
   GravaAlunos(l1);
   ...
l2 = LeAluno();
   ...


void GravaAlunos(LAluno l) {
  FILE *f;
   f= fopen("alunos.dat", "w");
   fwrite(&l.nalunos, sizeof(int), 1, f);   => aqui é quantos vou escrever
   fwrite(l.lista, sizeof(Aluno), l.nalunos, f);  => aqui vou escrevê-los todos de uma vez.
      é um array mas precisa do &
   fclose(f);
}
LAluno LeAlunos(){
  LAluno aux;
  FILE *f;
   f= fopen("alunos.dat", "r");
   fread(&aux.nalunos, sizeof(int), 1, f);
   fread(aux.lista, sizeof(Aluno), aux.nalunos, f);
      fclose(f);
      return aux;
}
```

```
void GravaAlunos2 (ListaAluno l) {
    FILE *f;
    f = fopen ("alunos.dat", "w");
    fwrite (l.lista, sizeof(Aluno), l.nalunos, f);   => só passa o nº de posições ocupadas
    fclose (f);
}

ListaAluno LeAlunos2 () {
    ListaAluno aux = {0, {{}}};
    FILE *f;
    f = fopen ("alunos.dat", "r");
    while ( fread ( &(aux.lista[aux.nalunos]), sizeof(Aluno), 1, f) )
        aux.nalunos++;
    fclose(f);
    return aux;
}

typedef struct sAluno {
    char *nome;
    char *numero;
} Aluno;

typedef struct sListaAluno {
    int nalunos;
    Aluno lista[100];
} ListaAluno;
```




```
int valida (char *frase)
```





Union

Expressão = Lista de elementos

Elemento = Int / Op

```
typedef char Operador;
typedef union u_Elemento
{
    int valor;
    Operador o;
} Elemento;

typedef struct sExp {
    int tipo;
    Elemento e;
    struct sExp *seg;
} *Exp, NExp;

void listar (Exp e){
    if(e) {
        listarElem (e->e, e->tipo);
        listar (e->seg);
    }
}

Void listarElem (Elemento Elem, int tipo){
    switch(tipo)
    {
        case VALOR: printf ("%d", elem.valor);
                    break;
        case OP : printf ("%c", elem.o);
    }
}
```

```
#define VALOR 101
#define OP 102
```