Árvores Binárias - Alocação Sequencial

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_SEQ.

```
1   typedef struct {
2      int info;
3      int left;
4      int right;
5      int father;
6   } NODE;
7   typedef struct{
8      int root;
9      int nodeFree;
10     NODE nodes[NUMNODES];/*#define NUMNODES 100*/
11  }ARV_BIN_SEQ;
12  void maketree(ARV_BIN_SEQ *, int);
13  void setleft(ARV_BIN_SEQ *, int, int);
14  void setright(ARV_BIN_SEQ *, int, int);
15  int info(ARV_BIN_SEQ *, int);
16  int left(ARV_BIN_SEQ *, int);
17  int right(ARV_BIN_SEQ *, int);
18  int father(ARV_BIN_SEQ *, int);
19  int brother(ARV_BIN_SEQ *, int);
20  int isleft(ARV_BIN_SEQ *, int);
21  int isright(ARV_BIN_SEQ *, int);
```

```c
void maketree(ARV_BIN_SEQ *t, int x)
{
    int i, ind;
    for (i=0; i<NUMNODES-1; i++)
        t->nodes[i].left = i+1;
    t->nodes[i].left = -1;
    t->nodeFree=0;
    ind = getNode(t);
    if (ind != -1) {
        t->nodes[ind].info = x;
        t->nodes[ind].left = -1;
        t->nodes[ind].right = -1;
        t->nodes[ind].father = -1;
        t->root = ind;
    } else {
        printf("Impossivel construir a arvore!");
        exit(1);
    }
}
```

```c
int getNode(ARV_BIN_SEQ *t)
{
    if (t->nodeFree != -1)
    {
        int i = t->nodeFree;
        t->nodeFree = t->nodes[t->nodeFree].left;
        return i;
    }
    else
        return -1;
}
void freeNode(ARV_BIN_SEQ *t, int node)
{
    t->nodes[node].left = t->nodeFree;
```

Árvores Binárias - Alocação Sequencial

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_SEQ.

```
1   typedef struct {
2      int info;
3      int left;
4      int right;
5      int father;
6   } NODE;
7   typedef struct{
8      int root;
9      int nodeFree;
10     NODE nodes[NUMNODES];/*#define NUMNODES 100*/
11  }ARV_BIN_SEQ;
12  void maketree(ARV_BIN_SEQ *, int);
13  void setleft(ARV_BIN_SEQ *, int, int);
14  void setright(ARV_BIN_SEQ *, int, int);
15  int info(ARV_BIN_SEQ *, int);
16  int left(ARV_BIN_SEQ *, int);
17  int right(ARV_BIN_SEQ *, int);
18  int father(ARV_BIN_SEQ *, int);
19  int brother(ARV_BIN_SEQ *, int);
20  int isleft(ARV_BIN_SEQ *, int);
21  int isright(ARV_BIN_SEQ *, int);
```

```c
void setleft(ARV_BIN_SEQ *t, int p, int x)
{
    int ind = getNode(t);
    if (ind != -1) {
        t->nodes[p].left = ind;
        t->nodes[ind].info = x;
        t->nodes[ind].left = -1;
        t->nodes[ind].right = -1;
        t->nodes[ind].father = p;
    } else {
        printf("Impossivel inserir filho a esquerda!");
        exit(2);
    }
}
```

Árvores Binárias - Alocação Sequencial

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_SEQ.

```
1   typedef struct {
2      int info;
3      int left;
4      int right;
5      int father;
6   } NODE;
7   typedef struct{
8      int root;
9      int nodeFree;
10     NODE nodes[NUMNODES];/*#define NUMNODES 100*/
11  }ARV_BIN_SEQ;
12  void maketree(ARV_BIN_SEQ *, int);
13  void setleft(ARV_BIN_SEQ *, int, int);
14  void setright(ARV_BIN_SEQ *, int, int);
15  int info(ARV_BIN_SEQ *, int);
16  int left(ARV_BIN_SEQ *, int);
17  int right(ARV_BIN_SEQ *, int);
18  int father(ARV_BIN_SEQ *, int);
19  int brother(ARV_BIN_SEQ *, int);
20  int isleft(ARV_BIN_SEQ *, int);
21  int isright(ARV_BIN_SEQ *, int);
```

```c
void setright(ARV_BIN_SEQ *t, int p, int x)
{
    int ind = getNode(t);
    if (ind != -1) {

        t->nodes[ind].info = x;
        t->nodes[ind].left = -1;
        t->nodes[ind].right = -1;
        t->nodes[ind].father = p;
    } else {
        printf("Impossivel inserir filho a direita!");
        exit(2);
    }
}
```

Árvores Binárias - Alocação Sequencial

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_SEQ.

```
1   typedef struct {
2      int info;
3      int left;
4      int right;
5      int father;
6   } NODE;
7   typedef struct{
8      int root;
9      int nodeFree;
10     NODE nodes[NUMNODES];/*#define NUMNODES 100*/
11  }ARV_BIN_SEQ;
12  void maketree(ARV_BIN_SEQ *, int);
13  void setleft(ARV_BIN_SEQ *, int, int);
14  void setright(ARV_BIN_SEQ *, int, int);
15  int info(ARV_BIN_SEQ *, int);
16  int left(ARV_BIN_SEQ *, int);
17  int right(ARV_BIN_SEQ *, int);
18  int father(ARV_BIN_SEQ *, int);
19  int brother(ARV_BIN_SEQ *, int);
20  int isleft(ARV_BIN_SEQ *, int);
21  int isright(ARV_BIN_SEQ *, int);
```

```c
int info(ARV_BIN_SEQ *t, int p)
{
    return t->nodes[p].info;
}
```

Árvores Binárias - Alocação Sequencial

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_SEQ.

```
1   typedef struct {
2      int info;
3      int left;
4      int right;
5      int father;
6   } NODE;
7   typedef struct{
8      int root;
9      int nodeFree;
10     NODE nodes[NUMNODES];/*#define NUMNODES 100*/
11  }ARV_BIN_SEQ;
12  void maketree(ARV_BIN_SEQ *, int);
13  void setleft(ARV_BIN_SEQ *, int, int);
14  void setright(ARV_BIN_SEQ *, int, int);
15  int info(ARV_BIN_SEQ *, int);
16  int left(ARV_BIN_SEQ *, int);
17  int right(ARV_BIN_SEQ *, int);
18  int father(ARV_BIN_SEQ *, int);
19  int brother(ARV_BIN_SEQ *, int);
20  int isleft(ARV_BIN_SEQ *, int);
21  int isright(ARV_BIN_SEQ *, int);
```

```c
int left(ARV_BIN_SEQ *t, int p)
{
    return t->nodes[p].left;
}
```

Árvores Binárias - Alocação Sequencial

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_SEQ.

```
1   typedef struct {
2      int info;
3      int left;
4      int right;
5      int father;
6   } NODE;
7   typedef struct{
8      int root;
9      int nodeFree;
10     NODE nodes[NUMNODES];/*#define NUMNODES 100*/
11  }ARV_BIN_SEQ;
12  void maketree(ARV_BIN_SEQ *, int);
13  void setleft(ARV_BIN_SEQ *, int, int);
14  void setright(ARV_BIN_SEQ *, int, int);
15  int info(ARV_BIN_SEQ *, int);
16  int left(ARV_BIN_SEQ *, int);
17  int right(ARV_BIN_SEQ *, int);
18  int father(ARV_BIN_SEQ *, int);
19  int brother(ARV_BIN_SEQ *, int);
20  int isleft(ARV_BIN_SEQ *, int);
21  int isright(ARV_BIN_SEQ *, int);
```

```c
int right(ARV_BIN_SEQ *t, int p)
{
    return t->nodes[p].right;
}
```

Árvores Binárias - Alocação Sequencial

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_SEQ.

```
1   typedef struct {
2      int info;
3      int left;
4      int right;
5      int father;
6   } NODE;
7   typedef struct{
8      int root;
9      int nodeFree;
10     NODE nodes[NUMNODES];/*#define NUMNODES 100*/
11  }ARV_BIN_SEQ;
12  void maketree(ARV_BIN_SEQ *, int);
13  void setleft(ARV_BIN_SEQ *, int, int);
14  void setright(ARV_BIN_SEQ *, int, int);
15  int info(ARV_BIN_SEQ *, int);
16  int left(ARV_BIN_SEQ *, int);
17  int right(ARV_BIN_SEQ *, int);
18  int father(ARV_BIN_SEQ *, int);
19  int brother(ARV_BIN_SEQ *, int);
20  int isleft(ARV_BIN_SEQ *, int);
21  int isright(ARV_BIN_SEQ *, int);
```

```c
int father(ARV_BIN_SEQ *t, int p)
{
    return t->nodes[p].father;
}
```

Árvores Binárias - Alocação Sequencial

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_SEQ.

```
1   typedef struct {
2      int info;
3      int left;
4      int right;
5      int father;
6   } NODE;
7   typedef struct{
8      int root;
9      int nodeFree;
10     NODE nodes[NUMNODES];/*#define NUMNODES 100*/
11  }ARV_BIN_SEQ;
12  void maketree(ARV_BIN_SEQ *, int);
13  void setleft(ARV_BIN_SEQ *, int, int);
14  void setright(ARV_BIN_SEQ *, int, int);
15  int info(ARV_BIN_SEQ *, int);
16  int left(ARV_BIN_SEQ *, int);
17  int right(ARV_BIN_SEQ *, int);
18  int father(ARV_BIN_SEQ *, int);
19  int brother(ARV_BIN_SEQ *, int);
20  int isleft(ARV_BIN_SEQ *, int);
21  int isright(ARV_BIN_SEQ *, int);
```

```c
int brother(ARV_BIN_SEQ *t, int p) {
    if (father(t, p) != -1) /*Se não for a raiz*/
        if (isleft(t, p))
            return right(t,father(t, p));
        else
            return t->nodes[t->nodes[p].father].left;
    return -1;
}
```

Árvores Binárias - Alocação Sequencial

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_SEQ.

```
1   typedef struct {
2      int info;
3      int left;
4      int right;
5      int father;
6   } NODE;
7   typedef struct{
8      int root;
9      int nodeFree;
10     NODE nodes[NUMNODES];/*#define NUMNODES 100*/
11  }ARV_BIN_SEQ;
12  void maketree(ARV_BIN_SEQ *, int);
13  void setleft(ARV_BIN_SEQ *, int, int);
14  void setright(ARV_BIN_SEQ *, int, int);
15  int info(ARV_BIN_SEQ *, int);
16  int left(ARV_BIN_SEQ *, int);
17  int right(ARV_BIN_SEQ *, int);
18  int father(ARV_BIN_SEQ *, int);
19  int brother(ARV_BIN_SEQ *, int);
20  int isleft(ARV_BIN_SEQ *, int);
21  int isright(ARV_BIN_SEQ *, int);
```

```c
int isleft(ARV_BIN_SEQ *t, int p) {
    int q = father(t, p);
    if (q == -1) /*Se for a raiz*/
        return (0);
    if (left(t,q) == p)
        return (1);
    return (0);
}
```

Árvores Binárias - Alocação Sequencial

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_SEQ.

```
1   typedef struct {
2      int info;
3      int left;
4      int right;
5      int father;
6   } NODE;
7   typedef struct{
8      int root;
9      int nodeFree;
10     NODE nodes[NUMNODES];/*#define NUMNODES 100*/
11  }ARV_BIN_SEQ;
12  void maketree(ARV_BIN_SEQ *, int);
13  void setleft(ARV_BIN_SEQ *, int, int);
14  void setright(ARV_BIN_SEQ *, int, int);
15  int info(ARV_BIN_SEQ *, int);
16  int left(ARV_BIN_SEQ *, int);
17  int right(ARV_BIN_SEQ *, int);
18  int father(ARV_BIN_SEQ *, int);
19  int brother(ARV_BIN_SEQ *, int);
20  int isleft(ARV_BIN_SEQ *, int);
21  int isright(ARV_BIN_SEQ *, int);
```

```c
int isright(ARV_BIN_SEQ *t, int p) {
    if (father(t, p) != -1)
        return (!isleft(t,p));
    return (0);
}
```

Árvore Binária
Alocação Encadeada
Exemplos de Aplicação

UNIVASF

Árvores Binárias - Alocação Encadeada

Existem formas de armazenamento estático alternativas à apresentada. Mas,visando otimizar a utilização da memória, temos a possibilidade de armazenar uma árvore binária dinamicamente.

Para tal, podemos definir a estrutura para os nós da seguinte forma:

```
1  typedef struct node
2  {
3      int info;
4      struct node *left;
5      struct node *right;
6      struct node *father;
7  } NODE;
```

UNIVASF

```c
typedef struct node
{
    int info;
    struct node *left;
    struct node *right;
    struct node *father;
} NODE;
typedef NODE * ARV_BIN_ENC;
void maketree(ARV_BIN_ENC *, int);
void setleft(ARV_BIN_ENC, int);
void setright(ARV_BIN_ENC, int);
int info(ARV_BIN_ENC);
ARV_BIN_ENC left(ARV_BIN_ENC);
ARV_BIN_ENC right(ARV_BIN_ENC);
ARV_BIN_ENC father(ARV_BIN_ENC);
ARV_BIN_ENC brother(ARV_BIN_ENC);
int isleft(ARV_BIN_ENC);
int isright(ARV_BIN_ENC);
```

Árvores Binárias - Alocação Encadeada

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_ENC.

```
 1   typedef struct node
 2   {
 3       int info;
 4       struct node *left;
 5       struct node *right;
 6       struct node *father;
 7   } NODE;
 8   typedef NODE * ARV_BIN_ENC;
 9   void maketree(ARV_BIN_ENC *, int);
10   void setleft(ARV_BIN_ENC, int);
11   void setright(ARV_BIN_ENC, int);
12   int info(ARV_BIN_ENC);
13   ARV_BIN_ENC left(ARV_BIN_ENC);
14   ARV_BIN_ENC right(ARV_BIN_ENC);
15   ARV_BIN_ENC father(ARV_BIN_ENC);
16   ARV_BIN_ENC brother(ARV_BIN_ENC);
17   int isleft(ARV_BIN_ENC);
18   int isright(ARV_BIN_ENC);
```

```c
void maketree(ARV_BIN_ENC *t, int x)
{
    *t  = (ARV_BIN_ENC) malloc (sizeof (NODE));
    if (!(*t))
    {
        printf("Erro! Nao existe memoria disponivel!");
        exit (1);
    }
    (*t)->info = x;
    (*t)->left = NULL;
    (*t)->right = NULL;
    (*t)->father = NULL;
}
```

Árvores Binárias - Alocação Encadeada

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_ENC.

```
1   typedef struct node
2   {
3       int info;
4       struct node *left;
5       struct node *right;
6       struct node *father;
7   } NODE;
8   typedef NODE * ARV_BIN_ENC;
9   void maketree(ARV_BIN_ENC *, int);
10  void setleft(ARV_BIN_ENC, int);
11  void setright(ARV_BIN_ENC, int);
12  int info(ARV_BIN_ENC);
13  ARV_BIN_ENC left(ARV_BIN_ENC);
14  ARV_BIN_ENC right(ARV_BIN_ENC);
15  ARV_BIN_ENC father(ARV_BIN_ENC);
16  ARV_BIN_ENC brother(ARV_BIN_ENC);
17  int isleft(ARV_BIN_ENC);
18  int isright(ARV_BIN_ENC);
```

Árvores Binárias - Alocação Encadeada

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_ENC.

```
1   typedef struct node
2   {
3       int info;
4       struct node *left;
5       struct node *right;
6       struct node *father;
7   } NODE;
8   typedef NODE * ARV_BIN_ENC;
9   void maketree(ARV_BIN_ENC *, int);
10  void setleft(ARV_BIN_ENC, int);
11  void setright(ARV_BIN_ENC, int);
12  int info(ARV_BIN_ENC);
13  ARV_BIN_ENC left(ARV_BIN_ENC);
14  ARV_BIN_ENC right(ARV_BIN_ENC);
15  ARV_BIN_ENC father(ARV_BIN_ENC);
16  ARV_BIN_ENC brother(ARV_BIN_ENC);
17  int isleft(ARV_BIN_ENC);
18  int isright(ARV_BIN_ENC);
```

Árvores Binárias - Alocação Encadeada

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_ENC.

```
 1   typedef struct node
 2   {
 3       int info;
 4       struct node *left;
 5       struct node *right;
 6       struct node *father;
 7   } NODE;
 8   typedef NODE * ARV_BIN_ENC;
 9   void maketree(ARV_BIN_ENC *, int);
10   void setleft(ARV_BIN_ENC, int);
11   void setright(ARV_BIN_ENC, int);
12   int info(ARV_BIN_ENC);
13   ARV_BIN_ENC left(ARV_BIN_ENC);
14   ARV_BIN_ENC right(ARV_BIN_ENC);
15   ARV_BIN_ENC father(ARV_BIN_ENC);
16   ARV_BIN_ENC brother(ARV_BIN_ENC);
17   int isleft(ARV_BIN_ENC);
18   int isright(ARV_BIN_ENC);
```

Árvores Binárias - Alocação Encadeada

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_ENC.

```
1   typedef struct node
2   {
3       int info;
4       struct node *left;
5       struct node *right;
6       struct node *father;
7   } NODE;
8   typedef NODE * ARV_BIN_ENC;
9   void maketree(ARV_BIN_ENC *, int);
10  void setleft(ARV_BIN_ENC, int);
11  void setright(ARV_BIN_ENC, int);
12  int info(ARV_BIN_ENC);
13  ARV_BIN_ENC left(ARV_BIN_ENC);
14  ARV_BIN_ENC right(ARV_BIN_ENC);
15  ARV_BIN_ENC father(ARV_BIN_ENC);
16  ARV_BIN_ENC brother(ARV_BIN_ENC);
17  int isleft(ARV_BIN_ENC);
18  int isright(ARV_BIN_ENC);
```

Árvores Binárias - Alocação Encadeada

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_ENC.

```
 1   typedef struct node
 2   {
 3       int info;
 4       struct node *left;
 5       struct node *right;
 6       struct node *father;
 7   } NODE;
 8   typedef NODE * ARV_BIN_ENC;
 9   void maketree(ARV_BIN_ENC *, int);
10   void setleft(ARV_BIN_ENC, int);
11   void setright(ARV_BIN_ENC, int);
12   int info(ARV_BIN_ENC);
13   ARV_BIN_ENC left(ARV_BIN_ENC);
14   ARV_BIN_ENC right(ARV_BIN_ENC);
15   ARV_BIN_ENC father(ARV_BIN_ENC);
16   ARV_BIN_ENC brother(ARV_BIN_ENC);
17   int isleft(ARV_BIN_ENC);
18   int isright(ARV_BIN_ENC);
```

Árvores Binárias - Alocação Encadeada

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_ENC.

```
 1   typedef struct node
 2   {
 3       int info;
 4       struct node *left;
 5       struct node *right;
 6       struct node *father;
 7   } NODE;
 8   typedef NODE * ARV_BIN_ENC;
 9   void maketree(ARV_BIN_ENC *, int);
10   void setleft(ARV_BIN_ENC, int);
11   void setright(ARV_BIN_ENC, int);
12   int info(ARV_BIN_ENC);
13   ARV_BIN_ENC left(ARV_BIN_ENC);
14   ARV_BIN_ENC right(ARV_BIN_ENC);
15   ARV_BIN_ENC father(ARV_BIN_ENC);
16   ARV_BIN_ENC brother(ARV_BIN_ENC);
17   int isleft(ARV_BIN_ENC);
18   int isright(ARV_BIN_ENC);
```

Árvores Binárias - Alocação Encadeada

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_ENC.

```
 1   typedef struct node
 2   {
 3       int info;
 4       struct node *left;
 5       struct node *right;
 6       struct node *father;
 7   } NODE;
 8   typedef NODE * ARV_BIN_ENC;
 9   void maketree(ARV_BIN_ENC *, int);
10   void setleft(ARV_BIN_ENC, int);
11   void setright(ARV_BIN_ENC, int);
12   int info(ARV_BIN_ENC);
13   ARV_BIN_ENC left(ARV_BIN_ENC);
14   ARV_BIN_ENC right(ARV_BIN_ENC);
15   ARV_BIN_ENC father(ARV_BIN_ENC);
16   ARV_BIN_ENC brother(ARV_BIN_ENC);
17   int isleft(ARV_BIN_ENC);
18   int isright(ARV_BIN_ENC);
```

Árvores Binárias - Alocação Encadeada

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_ENC.

```
 1   typedef struct node
 2   {
 3       int info;
 4       struct node *left;
 5       struct node *right;
 6       struct node *father;
 7   } NODE;
 8   typedef NODE * ARV_BIN_ENC;
 9   void maketree(ARV_BIN_ENC *, int);
10   void setleft(ARV_BIN_ENC, int);
11   void setright(ARV_BIN_ENC, int);
12   int info(ARV_BIN_ENC);
13   ARV_BIN_ENC left(ARV_BIN_ENC);
14   ARV_BIN_ENC right(ARV_BIN_ENC);
15   ARV_BIN_ENC father(ARV_BIN_ENC);
16   ARV_BIN_ENC brother(ARV_BIN_ENC);
17   int isleft(ARV_BIN_ENC);
18   int isright(ARV_BIN_ENC);
```

Árvores Binárias - Alocação Encadeada

Com base no que foi visto implemente as operações que compõem o TAD ARV_BIN_ENC.

```
 1   typedef struct node
 2   {
 3       int info;
 4       struct node *left;
 5       struct node *right;
 6       struct node *father;
 7   } NODE;
 8   typedef NODE * ARV_BIN_ENC;
 9   void maketree(ARV_BIN_ENC *, int);
10   void setleft(ARV_BIN_ENC, int);
11   void setright(ARV_BIN_ENC, int);
12   int info(ARV_BIN_ENC);
13   ARV_BIN_ENC left(ARV_BIN_ENC);
14   ARV_BIN_ENC right(ARV_BIN_ENC);
15   ARV_BIN_ENC father(ARV_BIN_ENC);
16   ARV_BIN_ENC brother(ARV_BIN_ENC);
17   int isleft(ARV_BIN_ENC);
18   int isright(ARV_BIN_ENC);
```