

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);
```

```
1 void setleft(ARV_BIN_SEQ *t, int p, int x)
2 {
3     int ind = getNode(t);
4     if (ind != -1) {
5         t->nodes[p].left = ind;
6         t->nodes[ind].info = x;
7         t->nodes[ind].left = -1;
8         t->nodes[ind].right = -1;
9         t->nodes[ind].father = p;
10    } else {
11        printf("Impossivel inserir filho a esquerda!");
12        exit(2);
13    }
14 }
```

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);
```

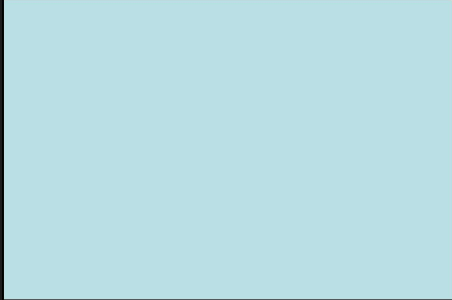
```
1 void setright(ARV_BIN_SEQ *t, int p, int x)
2 {
3     int ind = getNode(t);
4     if (ind != -1) {
5         t->nodes[p].right = ind;
6         t->nodes[ind].info = x;
7         t->nodes[ind].left = -1;
8         t->nodes[ind].right = -1;
9         t->nodes[ind].father = p;
10    } else {
11        printf("Impossivel inserir filho a direita!");
12        exit(2);
13    }
14 }
```

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);
```

```
1  int info(ARV_BIN_SEQ *t, int p)
2  {
3      return t->nodes[p].info;
4  }
```



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);
```

```
1  int left(ARV_BIN_SEQ *t, int p)
2  {
3      return t->nodes[p].left;
4  }
```

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);
```

```
1  int right(ARV_BIN_SEQ *t, int p)
2  {
3      return t->nodes[p].right;
4  }
```

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);
```

```
1 int father(ARV_BIN_SEQ *t, int p)
2 {
3     return t->nodes[p].father;
4 }
```

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);
```

```
1  int brother(ARV_BIN_SEQ *t, int p) {
2      if (father(t, p) != -1) /*Se não for a raiz*/
3          if (isleft(t, p))
4              return right(t, father(t, p));
5          else
6              return t->nodes[t->nodes[p].father].left;
7      return -1;
8  }
```

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);
```

```
1 int isleft(ARV_BIN_SEQ *t, int p) {
2     int q = father(t, p);
3     if (q == -1) /*Se for a raiz*/
4         return (0);
5     if (left(t,q) == p)
6         return (1);
7     return (0);
8 }
```

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);
```

```
1 int isright(ARV_BIN_SEQ *t, int p) {
2     if (father(t, p) != -1)
3         return (!isleft(t,p));
4     return (0);
5 }
```



Árvore Binária

Alocação Encadeada

Exemplos de Aplicação

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;
```

```
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);
```

```
1 void maketree(ARV_BIN_ENC *t, int x)
2 {
3     *t = (ARV_BIN_ENC) malloc (sizeof (NODE));
4     if (!(*t))
5     {
6         printf("Erro! Nao existe memoria disponivel!");
7         exit (1);
8     }
9     (*t)->info = x;
10    (*t)->left = NULL;
11    (*t)->right = NULL;
12    (*t)->father = NULL;
13 }
```

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);
```

```
1 void setleft(ARV_BIN_ENC t, int x)
2 {
3     t->left = (ARV_BIN_ENC) malloc (sizeof (NODE));
4     if (!(t->left))
5     {
6         printf("Erro! Nao existe memoria disponivel!");
7         exit (1);
8     }
9     t->left->info = x;
10    t->left->left = NULL;
11    t->left->right = NULL;
12    t->left->father = t;
13 }
```

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);
```

```
1 void setright(ARV_BIN_ENC t, int x)
2 {
3     t->right = (ARV_BIN_ENC) malloc (sizeof (NODE));
4     if (!(t->right))
5     {
6         printf("Erro! Nao existe memoria disponivel!");
7         exit (1);
8     }
9     t->right->info = x;
10    t->right->left = NULL;
11    t->right->right = NULL;
12    t->right->father = t;
13 }
```

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);
```

```
1  int info(ARV_BIN_ENC t)
2  {
3      return t->info;
4  }
```

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);
```

```
1  ARV_BIN_ENC left(ARV_BIN_ENC t)
2  {
3      return t->left;
4  }
```

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);
```

```
1  ARV_BIN_ENC right(ARV_BIN_ENC t)
2  {
3      return t->right;
4  }
```

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);
```

```
1  ARV_BIN_ENC father(ARV_BIN_ENC t)
2  {
3      return t->father;
4  }
```

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);
```

```
1  ARV_BIN_ENC brother(ARV_BIN_ENC t)
2  {
3      if (t->father)
4          if (isleft(t))
5              return (t->father)->right;
6          else
7              return (t->father)->left;
8      return NULL;
9  }
```

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);
```