1.2/Patricia.h

```c
/*---------------- File: Patricia.h  -------------------+
|TAD: Arvore Patricia                                  |
|                                                      |
| Do livro do Ziviani                                  |
| Adaptado por Guilherme C. Pena em 04/12/2023         |
+------------------------------------------------------+ */

#ifndef PATRICIA_H
#define PATRICIA_H

#include <stdio.h>
#include <stdlib.h>

#define D 6 // depende de TipoChave

typedef unsigned char TipoChave; // a definir, depende da aplicacao
typedef unsigned char TipoIndexAmp;

typedef enum {
    Interno,
    Externo
} TipoNo;

typedef struct patriciaNO *ArvorePat;

typedef struct patriciaNO {
    TipoNo nt; //NO type - Uma flag para dizer se o NO eh Externo ou Interno
    union {
        struct {
            TipoIndexAmp Index;
            ArvorePat Esq, Dir;
        } NInterno;
        TipoChave Chave;
    } NO;
} patriciaNO;

//Outra Tentativa da Funcao
int valorBit2(int i, TipoChave k){
    //Todo numero sera considerado com D bits
    //Exemplo: chave 5 em bin (101) -> (00000101)
    //Exemplo: chave 10 em bin (1010) -> (00001010)
    //Exemplo: chave 11 em bin (1011) -> (00001011)

    int n_bits = D;
    int p = 1, j, r;
    //int r = p << n_bits;//R teria o bit 8 ligado (10000000)

    // //Encontra o bit mais a esquerda primeiro
    // //Esse sera o bit 1
    // while((k & r) != r){ n_bits--; r = p << n_bits; }
    // n_bits++;

    //Desloca i bits a partir da esquerda
    for(j=1; j<=i; j++)
        n_bits--;

    if(n_bits < 0) return 0;
```

```
58      r = p << n_bits;
59
60      return ((k & r) == r);//Retorna se o i-esimo bit eh 1 ou 0
61  }
62
63  int retornaNBits(TipoChave k){
64      int n_bits = D;
65      int p = 1;
66      int r = p << n_bits;
67      //Encontra o bit mais a esquerda primeiro
68      //Esse sera o bit 1
69      while((k & r) != r){ n_bits--; r = p << n_bits; }
70      n_bits++;
71      return n_bits;
72  }
73
74  int valorBit(int i, TipoChave k){
75
76      int n_bits = D;
77      int p = 1, j;
78      int r = p << n_bits;
79      //Encontra o bit mais a esquerda primeiro
80      //Esse sera o bit 1
81      while((k & r) != r){ n_bits--; r = p << n_bits; }
82      n_bits++;
83
84
85
86      //Desloca i bits a partir da esquerda
87      for(j=1; j<=i; j++)
88          n_bits--;
89
90      if(n_bits < 0) return 0;
91      r = p << n_bits;
92
93      //printf("Bit[%d] em %d: %d\n", i, k, (k & r) == r);
94
95      return ((k & r) == r);//Retorna se o i-esimo bit eh 1 ou 0
96  }
97
98  int EExterno(ArvorePat p) {
99      // Verifica se p^ eh um nodo externo
100     return (p->nt == Externo);
101 }
102
103 ArvorePat CriaNoInt(int i, ArvorePat *Esq, ArvorePat *Dir) {
104     ArvorePat p;
105     p = (ArvorePat)malloc(sizeof(patriciaNO));
106     p->nt = Interno;
107     p->NO.NInterno.Esq = *Esq;
108     p->NO.NInterno.Dir = *Dir;
109     p->NO.NInterno.Index = i;
110     return p;
111 }
112
113 ArvorePat CriaNoExt(TipoChave k) {
114     ArvorePat p;
115     p = (ArvorePat)malloc(sizeof(patriciaNO));
116     p->nt = Externo;
117     p->NO.Chave = k;
```

```
118     return p;
119 }
120 
121 int Pesquisa(TipoChave k, ArvorePat t) {
122     if (EExterno(t)) {
123         if (k == t->NO.Chave){
124             printf("Elemento %d encontrado!\n", k);
125             return 1;
126         }
127         else{
128             printf("Elemento %d NAO encontrado!\n", k);
129             return 0;
130         }
131     }
132     if (valorBit2(t->NO.NInterno.Index, k) == 0)
133         return Pesquisa(k, t->NO.NInterno.Esq);
134     else
135         return Pesquisa(k, t->NO.NInterno.Dir);
136 }
137 
138 ArvorePat InsereEntre(TipoChave k, ArvorePat *t, int i) {
139     //printf("Insere Entre %d, [%d]\n", k, i);
140     ArvorePat p;
141     if (EExterno(*t) || i < (*t)->NO.NInterno.Index) {
142 
143         int nb1 = retornaNBits((*t)->NO.Chave);
144         int nb2 = retornaNBits(k);
145         // cria um novo no externo
146         p = CriaNoExt(k);
147         if(nb1 <= nb2){
148             if (valorBit2(i, k) == 1)
149                 return CriaNoInt(i, t, &p);
150             else
151                 return CriaNoInt(i, &p, t);
152         }else{
153             return CriaNoInt(i, &p, t);
154         }
155     } else {
156         if (valorBit2((*t)->NO.NInterno.Index, k) == 1)
157             (*t)->NO.NInterno.Dir = InsereEntre(k, &((*t)->NO.NInterno.Dir), i);
158         else
159             (*t)->NO.NInterno.Esq = InsereEntre(k, &((*t)->NO.NInterno.Esq), i);
160         return (*t);
161     }
162 }
163 
164 ArvorePat Insere(TipoChave k, ArvorePat *t) {
165     printf("Inserindo %d..\n", k);
166     ArvorePat p;
167     int i;
168     if (*t == NULL)
169         return CriaNoExt(k);
170     else {
171         if(Pesquisa(k, *t)){
172             printf("Erro: chave ja esta na arvore\n");
173             return (*t);
174         }
175         p = *t;
176         while (!EExterno(p)) {
177             if (valorBit2(p->NO.NInterno.Index, k) == 1)
```

```
178                 p = p->NO.NInterno.Dir;
179             else
180                 p = p->NO.NInterno.Esq;
181         }
182         // acha o primeiro bit diferente
183         i = 1;
184         while ((i <= D) && ( valorBit2((int)i, k) == valorBit2((int)i, p->
    NO.Chave) ) )
185             i++;
186         printf("Bit diferente eh: [%d]\n", i);
187         if (i > D) {
188             //fazer outra coisa
189             //No caso de exemplo, inserir 5 (Bin: 101) antes de 10 (Bin: 1010),
    funciona corretamente
190             //No entanto, ainda nao funciona se inserer 10 antes de 5
191             return InsereEntre(k, t, 1);
192         } else
193             return InsereEntre(k, t, i);
194     }
195 }
196
197
198 void pre_ordem(ArvorePat raiz, int nivel){
199     if(raiz != NULL){
200         printf("Nivel %d: ", nivel);
201         if(raiz->nt == Interno){
202             printf("(INT) %d\n", raiz->NO.NInterno.Index);
203             pre_ordem(raiz->NO.NInterno.Esq, nivel+1);
204             pre_ordem(raiz->NO.NInterno.Dir, nivel+1);
205         }else{
206             printf("(EXT) %d\n", raiz->NO.Chave);
207         }
208     }
209 }
210
211 void imprimePatricia(ArvorePat raiz){
212     if(raiz == NULL) return;
213     pre_ordem(raiz, 0);
214     printf("\n");
215 }
216
217 #endif
```