INF 1010

Estruturas de Dados Avançadas

Tipo abstrato de dados

Revisão – estrutura de dados e tipo abstrato de dados

programa
que usa esse
modulo

#include “modulo.h"

interface: modulo.h

modulo.c
implementação

aqui aparecem as estruturas de dados variadas!

programa
que usa conjunto

```
#include “conjunto.h”

...
int main (void) {
  tconjunto C1, C2;
  ...
  C1 = criaConj();
  ...
```

interface: conjunto.h

```
typedef struct
      sconjunto *tconjunto;

tconjunto criaConj (void);
int insereConj (tconjunto c,
       int e);
...
int estaNoConj (tconjunto c,
         int e);
...
```

conjunto.c
implementação

lista
encadeada?
array?

Módulo Conjunto - interface

```
typedef struct
      sconjunto *tconjunto;

tconjunto criaConj (void);

int eConjValido (tconjunto c);

tconjunto insereConj (tconjunto c, int e);

tconjunto retiraConj (tconjunto c, int e);

int estaNoConj (tconjunto c, int e);

/* ... outras */
```

Módulo Conjunto - 1

```
#include "conjunto.h"
#include <stdlib.h>
#include <stdio.h>

struct sconjunto {
  int info;
  struct sconjunto *prox;
};

tconjunto criaConj (void) {
  return NULL;
}

...
```

Módulo Conjunto - 1

```
#include "conjunto.h”  /* fundamental */
#include <stdlib.h>
#include <stdio.h>

struct sconjunto {
  int info;
  struct sconjunto *prox;
};

tconjunto criaConj (void) {
  return NULL;
}

...
```

Módulo Conjunto – 2

```
...
tconjunto insereConj (tconjunto c, int e) {

  struct sconjunto *novo =
                          (struct sconjunto *) malloc (sizeof(struct sconjunto));
  novo->info = e;
  novo->prox = c;
  return novo;
}

int estaNoConj (tconjunto c, int e) {
   while (c!=NULL)
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;
}
```

Módulo Conjunto – 2

```
...
tconjunto insereConj (tconjunto c, int e) {
  /* insere no início do conjunto */
  struct sconjunto *novo =
                          (struct sconjunto *) malloc (sizeof(struct sconjunto));
  novo->info = e;  /* e se não conseguiu alocar??? seg fault! */
  novo->prox = c;
  return novo;
}

int estaNoConj (tconjunto c, int e) {
   while (c!=NULL)
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;
}
```

Módulo Conjunto – 2

```
...
tconjunto insereConj (tconjunto c, int e) {

  struct sconjunto *novo =
                          (struct sconjunto *) malloc (sizeof(struct sconjunto));
  if (novo!=NULL) {
    novo->info = e;
    novo->prox = c;
  }
  return novo;
}

int estaNoConj (tconjunto c, int e) {
   while (c!=NULL)
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;
}
```

Módulo Conjunto – 2

```
...
tconjunto insereConj (tconjunto c, int e) {
struct sconjunto *novo =
                          (struct sconjunto *) malloc (sizeof(struct sconjunto));
  if (novo!=NULL) {
    novo->info = e;
    novo->prox = c;
  }
  return novo; /* se não conseguiu inserir vai retornar NULL */
}

int estaNoConj (tconjunto c, int e) {
   while (c!=NULL)
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;
}
```

Problemas dessa implementação

- para eliminar...

Elemento já pertence ao conjunto?

Para eliminar é necessário percorrer todo o conjunto (podem ter elementos iguais, nao verifica se elemento já foi incluído).

Módulo Conjunto – 2.1

```
...
tconjunto insereConj (tconjunto c, int e) {
  if (estaNoConj (c, e)) return c;
  struct sconjunto *novo =
                          (struct sconjunto *) malloc (sizeof(struct sconjunto));
  if (novo!=NULL) {
    novo->info = e;
    novo->prox = c;
  }
  return novo;
}

int estaNoConj (tconjunto c, int e) {
   while (c!=NULL)
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;
}
```

Problemas dessa implementação

- Para saber que está no conjunto…

  Percorrer a lista inteira

Módulo Conjunto – outra implementação

```
tconjunto insereConj (tconjunto c, int e) { /* lista ordenada */

  tconjunto novo;

  if (estaNoConj (c, e)) return c;

  novo = (struct sconjunto *) malloc (sizeof(struct sconjunto));
  if (novo!=NULL) novo->info = e
  else return NULL;

  /* tratar o caso qdo ainda não tem elemento algum: */

  /* caso geral: */
  /* percorrer lista até encontrar posição correta */
}
```

```
tconjunto insereConj (tconjunto c, int e) {

 …
/* percorrer até achar a posição certa! */
  aux1 = c;
  aux2 = c->prox;
  while (1)
    if ((aux2==NULL) || (aux2->info > e)) {
      /* achou posicao */
      novo->prox = aux2;
      aux1->prox = novo;
      return c;
    }
    else {
      aux1 = aux2;
      aux2 = aux2->prox;
    }
}
```

Módulo Conjunto – outra implementação

```
int estaNoConj (tconjunto c, int e) {

   while ((c!=NULL) && (c->info <= e))
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;

}
```

... e se não é um conjunto, mas sim um MAPA?

mapeamento (chave, valor)

Módulo Mapa- interface

```
typedef struct smapa *tmapa;

tmapa criaMapa (void);

int eMapa (tmapa c);

tmapa insereMapa (tmapa c, int chave, int valor);

tmapa retiraMapa (tmapa c, int chave);

int estaNoMapa (tmapa c, int chave);

/* ... outras */
```

... e se o elemento é mais complexo que um inteiro?

Módulo Mapa de objetos - interface

```
#include "meuobjeto.h"

typedef struct smapa *tmapa;

tmapa criaMapa (void);

int eMapa (tmapa c);

tmapa insereMapa (tmapa c, int chave, tobjeto *);

tmapa retiraMapa (tmapa c, int chave);

int estaNoMapa (tmapa c, int chave);

/* ... outras */
```

Módulo Mapa – Implementação

```
#include "mapaObj.h"
#include <stdlib.h>
#include <stdio.h>

struct smapa {
  int chave;
  tobjeto* obj;
  struct smapa *prox;
};

tmapa criaMapa (void) {
  return NULL;
}

...
```

Perguntas?