Exercício: TAD Conjuntos (SET)

- Um conjunto é uma coleção de membros (ou elementos); cada membro ou é um conjunto ou um elemento primitivo chamado de átomo.
- Todos os membros são diferentes: nenhum conjunto contém 2 cópias do mesmo elemento.
- Ex:

{1,4} ok

{1,4,1} não ok

Operações básicas: união, intersecção e diferença

- Se A e B são conjuntos, então A ∪ B é o conjunto de elementos que são membros de A ou de B ou de ambos
- Se A e B são conjuntos, então A ∩ B é o conjunto de elementos que estão em A e em B
- Se A e B são conjuntos, então A – B é o conjunto de elementos em A que não estão em B
- Exemplo: A = {a,b,c} B = {b,d}
  A ∪ B =
  A ∩ B =
  A – B =

Operações básicas: união, intersecção e diferença

- Se A e B são conjuntos, então A ∪ B é o conjunto de elementos que são membros de A ou de B ou de ambos
- Se A e B são conjuntos, então A ∩ B é o conjunto de elementos que estão em A e em B
- Se A e B são conjuntos, então A – B é o conjunto de elementos em A que não estão em B
- Exemplo: A = {a,b,c} B = {b,d}
  A ∪ B = {a,b,c,d}
  A ∩ B = {b}
  A – B = {a,c}

Conjuntos em C

- Como implementar um conjunto em C?

Conjuntos em C

- Como implementar um conjunto em C?

```
# define N 100 //por exemplo, conjunto que tem números de 0 a 99
int conjunto[N]; //conjunto[i]=1 se i está no conjunto; 0, caso contrário
```


Operações?

Operações usuais

- União(A,B,C)
- Intersecção(A,B,C)
- Diferença(A,B,C)
- Membro(x,A)
- Cria_Conj_Vazio(A)
- Insere(x,A)
- Remove(x,A)
- Atribui(A,B)
- Min(A)
- Max(A)
- Igual(A,B)

Definição das operações

- União(A,B,C): toma os argumentos A e B que são conjuntos e retorna A ∪ B à variavel C
- Intersecção(A,B,C): toma os argumentos A e B que são conjuntos e retorna A ∩ B à variavel C
- Diferença(A,B,C): toma os argumentos A e B que são conjuntos e retorna A - B à variavel C
- Membro(x,A): toma o conjunto A e o objeto x cujo tipo é o tipo do elemento de A e retorna um valor booleano – true se x ∈ A e false caso contrário
- Cria_Conj_Vazio(A): faz o conjunto vazio ser o valor para a variável conjunto A

Definição das operações

- Insere(x,A): toma o conjunto A e o objeto x cujo tipo é o tipo do elemento de A e faz x um membro de A. O novo valor de A = A ∪ {x}. Se x já é um membro de A, então a operação insere não muda A
- Remove(x,A): remove o objeto x, cujo tipo é o tipo do elemento de A, de A. O novo valor de A = A – {x}. Se x não pertence a A então a operação remove não altera A

Definição das operações

- Atribui(A,B): Seta o valor da variável conjunto A = ao valor da variável conjunto B
- Min(A): retorna o valor mínimo no conjunto A. Por exemplo: Min({2,3,1}) = 1 e Min({‘a’,’b’,’c’}) = ‘a’
- Max(A): Similar a Min(A) só que retorna o máximo do conjunto
- Igual(A,B): retorna true se e somente se os conjuntos A e B consistem dos mesmos elementos

Exercício

- Em duplas, implemente em C o TAD conjunto de números inteiros
  - Use um arquivo .h

TADs em C: Exemplo

```
/* TAD: conjunto*/

/* Tipo Exportado */
typedef struct conjunto Conjunto;

/* Funções Exportadas */


/* Função união - Une os elementos do conjunto A e B em um
   conjunto C.Retorna 1 se a operação for bem sucedida e 0 caso
   contrario */
int uniao (Conjunto *A, Conjunto *B, Conjunto *C);
/* Função Intersecção - Armazena em C os mesmos elementos que
   estão no conjunto A e B*/
void interseccao (Conjunto *A, Conjunto *B, Conjunto *C);
/* Função libera - Libera a memória de um conjunto */
void libera (Conjunto *A);
```

Arquivo conjunto.h

TADs em C: Exemplo

```
/* Continuação... */
/* Função Cria_Conj_Vazio - Cria um conjunto vazio e retorna o conjunto
   criado. A variável erro retorna 1 se o conjunto foi criado corretamente
   e 0 caso contrario. Deve ser usado como primeira operação sobre um
   conjunto. */
Conjunto *Cria_Conj_Vazio(int *erro);

/* Função diferenca - atribui ao conjunto C a diferença entre os
   conjuntos A e B  */
void diferenca(Conjunto *A, Conjunto *B, Conjunto *C);

/* Função membro - verifica se o elemento elem está no Conjunto
   A   */
int membro(elem x, Conjunto *A);

/* Função insere - insere o elemento elem no conjunto A e
   retorna se a execução foi realizada com sucesso(1) ou
   não(0)*/
int insere(elem x, Conjunto *A);
```

30

Arquivo conjunto.h

TADs em C: Exemplo

```
/* Continuação... */
/* Função remove - remove o elemento elem do Conjunto A,
   retorna 1 se o elemento foi retirado e 0 se o elemento
   não está no conjunto */
int remove(elem x, Conjunto *A);
/* Função Atribui - faz a copia do conjunto A para o B*/
void atribui(Conjunto *A, Conjunto *B);
/* Função min - retorna o menor elemento do conjunto A -
   se o conjunto está vazio retorna TAM */
elem min(Conjunto *A);
```

Arquivo conjunto.h

31

TADs em C: Exemplo

```
/* Continuação... */

/* Função max - retorna o maior elemento do conjunto A - se o
   conjunto está vazio retorna TAM */
elem max(Conjunto *A);
/* Função igual - verifica se o conjunto A é igual a
Conjunto B   */
int igual(Conjunto *A, Conjunto *B);
/* Função tamanho - retorna o tamanho do conjunto A */
int tamanho(Conjunto *A);
/* Função testa_vazio - verifica se o conjunto A é vazio 1 se
   for vazio e 0 caso contrario*/
int testa_vazio(Conjunto *A);
```

32

Arquivo conjunto.h

TADs em C: Exemplo

```
#include <stdlib.h> /* malloc, free, exit */
#include <stdio.h> /* printf */
#include "conjunto.h"

#define TAM 100 // trabalha com elementos do universo 0..TAM -1
typedef int elem;

struct conjunto {
        elem* v; //vetor booleano que armazenará o conjunto sendo que
         //o índice armazena o valor sendo true se o elemento está
         // no conjunto, false caso contrário
};

void libera (Conjunto *A){
   free(A->v);
   free(A);
}
```

33

Arquivo conjunto.c

TADs em C: Exemplo

```
/* Continuação... */

int uniao (Conjunto *A, Conjunto *B, Conjunto *C) {
   int i;// variável auxiliar para realização do loop
   for(i = 0; i<TAM;i++){
     if (A->v[i]==1) || (B->v[i]==1) {
        C->v[i]=1;
        }
   }
  return 1; /*na implementação com vetores de booleanos sempre é possível
   realizar a união, pois o conjunto de elementos é limitado a um universo*/

}
void interseccao(Conjunto *A, Conjunto *B, Conjunto *C) {
   int i; // variável auxiliar para realização do loop

   for(i = 0; i<TAM;i++){
     if (A->v[i]==1) && (B->v[i]==1) {
        C->v[i]=1;
        }
}
```

Arquivo conjunto.c

TADs em C: Exemplo

```
/* Continuação... */
/* faz o conjunto vazio ser o valor para a variável conjunto A. Deve ser
   usado como primeira operação sobre um conjunto.*/
Conjunto* Cria_Conj_Vazio(int *erro) {
   int i; // variável auxiliar para realização do loop
   Conjunto* conj = (Conjunto*) malloc(sizeof(Conjunto));
   if (conj == NULL) {
        erro= 1;
        exit(1);
   }
   conj->v = (int*)malloc(TAM*sizeof(int));
   if (conj->v == NULL) {
        erro= 1;
        exit(1);
   }
   for(i=0;i<TAM;i++){
     conj->v[i]=0;
   }
   erro = 0;
   return conj;
}
```

Arquivo conjunto.c

TADs em C: Exemplo

```
/* Continuação... */
int insere(elem x, Conjunto *A) {
   if (x>=TAM || x<0 ) {
        return 0;
   }
   A->v[x]=1;
   return 1;
}

int membro(elem x, Conjunto *A) {
   if (x>=TAM || x<0 ||(A->v[x]==0)) {
     return 0;
   }
   return 1;
}
```

Arquivo conjunto.c

TADs em C: Exemplo

```
/* Continuação... */
int remove(elem x, Conjunto *A) {
   if (x>=TAM || x<0 ||(A->v[x]==0)) {
        return 0;
   }
   A->v[x]=0;
   return 1;
}

void atribui(Conjunto *A, Conjunto *B) {
   int i; // variável auxiliar para realização do loop
   for(i = 0; i<TAM;i++){
     B->v[i]=A->v[i];
   }
}
```

Arquivo conjunto.c

TADs em C: Exemplo

```
/* Continuação... */

elem min (Conjunto *A) {
   int i;
   for(i = 0; i<TAM;i++){
     if (A->v[i]==1) {
        return i;
     }
   }
   return TAM; // condição de conjunto vazio
}

elem max (Conjunto *A) {
   int i;
   for(i = TAM - 1; i>-1;i--){
     if (A->v[i]==1) {
        return i;
     }
   }
   return TAM; // condição de conjunto vazio
}
```

Arquivo conjunto.c

TADs em C: Exemplo

```
/* Continuação... */

void diferenca(Conjunto *A, Conjunto *B, Conjunto *C){
   int i; // variável auxiliar para realização do loop

   for(i = 0; i<TAM;i++){
     if (A->v[i]==1) && (B->v[i]==0) {
        C->v[i]=1;
        }
}

int igual(Conjunto *A, Conjunto *B){
   int i;
   for(i = 0; i<TAM;i++){
     if (A->v[i]!=B->v[i]) {
        return 0;
     }
   }
   return 1;
}
```

Arquivo conjunto.c

TADs em C: Exemplo

```
/* Continuação... */

int tamanho(Conjunto *A){
   int i,tam; // variável auxiliar para realização do loop e para a
   // verificação do tamanho do conjunto
   tam = 0;
   for(i = 0; i<TAM;i++){
     if (A->v[i]==1) {
       tam++;
     }
   }
   return tam;
}

int testa_vazio(Conjunto *A);{
   int i;
   for(i = 0; i<TAM;i++){
     if (A->v[i]==1) {
        return 0;
     }
   }
   return 1;
}
```

Arquivo conjunto.c

Tipo abstrato de dados

- *Pensamento do dia*

Nunca desmonte uma TV para aumentar o volume. Use o botão!