# Algoritmos e Estruturas de Dados I

Profa. Sofia Costa Paiva - 2023/1

Aula 6 – TAD Conjuntos

INF
INSTITUTO DE INFORMÁTICA
UFG
UNIVERSIDADE FEDERAL DE GOIÁS

# Exercício: TAD Conjuntos

- Um conjunto é uma coleção de membros (ou elementos)
  - Cada membro ou é um conjunto ou um elemento primitivo chamado de átomo
    - Considerem apenas elementos do tipo átomo
- Todos os membros são diferentes: nenhum conjunto contém 2 cópias do mesmo elemento
- Exemplo
    - {1,4} -> ok
    - {1,4,1} -> não ok

# Operações básicas

- Se A e B são conjuntos, então $A \cup B$ é o conjunto de elementos que são membros de A ou de B ou de ambos
- Se A e B são conjuntos, então $A \cap B$ 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: $A = \{a,b,c\}$ e $B = \{b,d\}$

  $A \cup B = \{a,b,c,d\}$

  $A \cap B = \{b\}$

  $A - B = \{a,c\}$

# Conjuntos em C

- Como implementar um conjunto em C?

# Conjuntos em C

- Como implementar um conjunto em C?

```
char conjunto[1000]; //elementos são concatenados com separadores
```

# Conjuntos em C

- Como implementar um conjunto em C?

```
char conjunto[1000]; //elementos são concatenados com separadores

int e1, e2, ..., eN; //uma variável para cada elemento
```

# Conjuntos em C

- Como implementar um conjunto em C?

```
char conjunto[1000]; //elementos são concatenados com separadores

int e1, e2, ..., eN; //uma variável para cada elemento

# 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
```

# Conjuntos em C

- Como implementar um conjunto em C?

```
char conjunto[1000]; //elementos são concatenados com separadores

int e1, e2, ..., eN; //uma variável para cada elemento

# 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

struct conjunto { //uma estrutura dinâmica
int* v;
}
v=(int*) malloc(sizeof(int)*100);
```

# Operações?

# Operações usuais

- Criar_conjunto(A)
- União(A,B,C)
- Intersecção(A,B,C)
- Diferença(A,B,C)
- Membro(x,A)
- Inserir(x,A)
- Remover(x,A)
- Atribuir(A,B)
- Min(A) e Max(A)
- Igual(A,B)
- Liberar(A)
- Imprimir(A)

# Definição das operações

- Criar_conjunto(A): faz o conjunto vazio ser o valor para a variável conjunto A
- União(A,B,C): toma os argumentos A e B que são conjuntos e retorna A ∪ B na variável C
- Intersecção(A,B,C): toma os argumentos A e B que são conjuntos e retorna A ∩ B na variável C
- Diferença(A,B,C): toma os argumentos A e B que são conjuntos e retorna A - B na variável 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

# Definição das operações

- Inserir(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 não muda A

- Remover(x,A): remove de A o objeto x, cujo tipo é o tipo do elemento de A. O novo valor de A = A – {x}. Se x não pertence a A, então a operação não altera A

# Definição das operações

- Atribuir(A,B): seta o valor da variável conjunto A igual 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
- Liberar(A): libera memória usada por A
- Imprimir(A): imprime elementos do conjunto A

# Tipo abstrato de dados

- *Pensamento do dia*

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