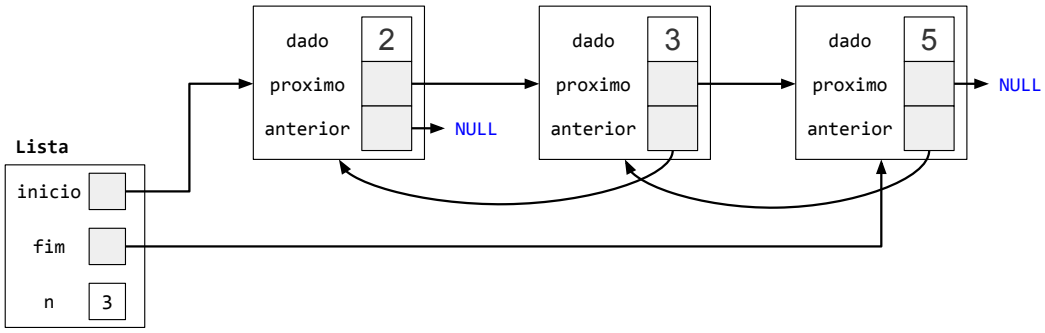
Universidade Federal do Espírito Santo

Departamento de Computação e Eletrônica - CEUNES
Estrutura de Dados I
Prof. Oberlan Romão

Listas Duplamente Encadeadas

Atenção: Ao terminar, não se esqueça de enviar as soluções no AVA.

Em uma lista duplamente encadeada (ilustrada na figura abaixo), cada nó tem um ponteiro para o próximo nó e um ponteiro para o nó anterior. Desta forma, dado um nó, podemos acessar ambos os elementos adjacentes: o próximo e o anterior. Assim como em outras estruturas, devemos ter o cuidado de manter os apontadores da lista e dos nós sempre consistentes.



Nessa aula prática, sua tarefa é implementar/completar algumas função de um TAD Conjunto que utiliza (internamente) uma lista duplamente encadeada.

TAD Conjunto

Em matemática, um conjunto é uma coleção de elementos **distintos**. Se a é um elemento de um conjunto A, escrevemos $a \in A$ (leia-se: "a é um elemento de A" ou "a pertence a A"), enquanto que $a \notin A$ significa que a não é elemento de A.

Como em um conjunto não pode haver elementos repetidos, devemos ter o cuidado de não permitir a inserção de um elemento que já está no conjunto. Dessa forma, é crucial que sejam utilizadas estruturas eficientes para decidir se devemos ou não incluir um determinado elemento no conjunto. Usando uma estrutura do tipo lista, sabemos que uma função que procura um elemento possui complexidade $O(n)$. Nas próximas aulas estudaremos estruturas mais eficientes :)

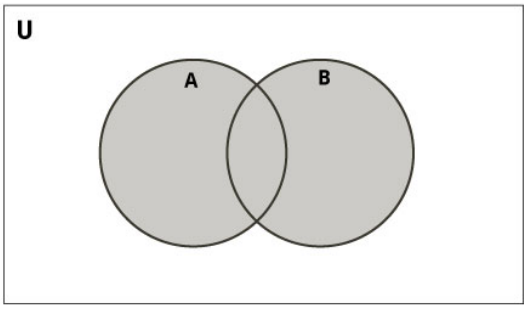
Baixe, no AVA, o arquivo `Conjunto.zip`, extraia em alguma pasta e analise todo o código. Veja que há algumas função em `Conjunto.c` que usam algumas novidades (em especial a função `criaConjuntoValores`). Várias função não estão implementadas ou estão incompletas. Vamos, por parte, implementar cada uma. Para cada um dos exercícios abaixo você deve descomentar, na função `main()`, a chamada da função associada ao exercício. Você pode comparar a sua solução com a saída esperada (`SaidaEsperada.txt`).

Exercício 1: No terminal, digite `make run`. Deu tudo certo? O que está faltando na função `insere`? Corrija-a.
Exercício 2: Na função `main()`, descomente a linha da chamada da função `exercicio2()`. Analise o código dessa função para entender quais funções do TAD Conjunto ela utiliza. Complete a função necessária para que a função `exercicio2()` funcione corretamente.
Exercício 3: Complete a função `Conjunto *copia(Conjunto *A)`. Ela deve retornar uma cópia independente do conjunto passado como parâmetro.
Exercício 4: Implemente a função `Conjunto *uniao(Conjunto *A, Conjunto *B)`. A função recebe dois conjuntos e retorna outro conjunto contendo a união dos conjuntos passado como argumentos. Lembrando que a união de dois conjuntos quaisquer A e B é o conjunto

$$A \cup B = \{x : x \in A \text{ ou } x \in B\}$$

Exemplos:

- $\{1, 3, 5, 7, 9\} \cup \{2, 4, 6, 8\} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $\{1, 2, 4, 8\} \cup \{2, 3, 5, 7\} = \{1, 2, 3, 4, 5, 7, 8\}$
- $\{2, 3\} \cup \{1, 2, 3, 4\} = \{1, 2, 3, 4\}$

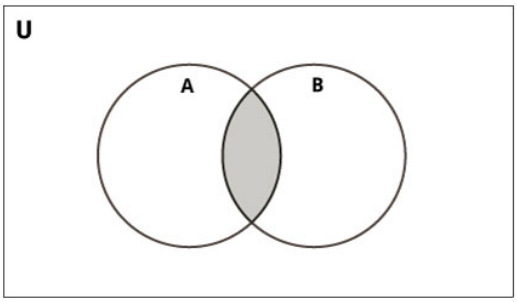


Exercício 5: Faça a função `Conjunto *intersecao(Conjunto *A, Conjunto *B)`. A função recebe dois conjuntos e retorna outro conjunto contendo a interseção dos conjuntos passado como argumentos. Lembrando que a interseção de dois conjuntos quaisquer A e B é o conjunto

$$A \cap B = \{x : x \in A \text{ e } x \in B\}$$

Se $A \cap B = \varnothing$, dizemos que A e B são **conjuntos disjuntos**.
Exemplos:

- $\{1, 3, 5, 7, 9\} \cap \{2, 4, 6, 8\} = \varnothing$
- $\{1, 2, 4, 8\} \cap \{2, 3, 5, 7\} = \{2\}$
- $\{2, 3\} \cap \{1, 2, 3, 4\} = \{2, 3\}$

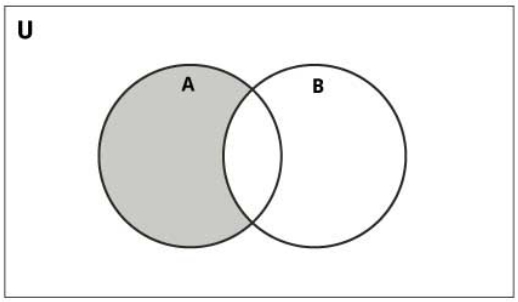


Exercício 6: Complete a função `Conjunto *diferenca(Conjunto *A, Conjunto *B)`. A função recebe dois conjuntos e retorna outro conjunto contendo a diferença dos conjuntos passado como argumentos. Lembrando que a diferença de dois conjuntos quaisquer A e B é o conjunto

$$A - B = \{x : x \in A \text{ e } x \notin B\}$$

Exemplos:

- $\{1, 3, 5, 7, 9\} - \{2, 4, 6, 8\} = \{1, 3, 5, 7, 9\}$
- $\{1, 2, 4, 8\} - \{2, 3, 5, 7\} = \{1, 4, 8\}$
- $\{2, 3\} - \{1, 2, 3, 4\} = \varnothing$



Exercício 7: Complete a função `bool iguais(Conjunto *A, Conjunto *B)`. A função recebe dois conjuntos e retorna `true` se os dois conjuntos forem iguais ou `false`, caso contrário. Lembrando que dois conjuntos A e B são iguais ou idênticos quando contém os mesmos elementos. Isto é, $A = B$ significa

$$(\forall x)[(x \in A) \leftrightarrow (x \in B)].$$

Exercício 8: Na função desse exercício, é utilizado uma função de função de alta ordem (ou seja, uma função que recebe outra função como parâmetro) que conta os elementos do conjunto que satisfazem a função de comparação (passada como parâmetro). Analise como devemos chamar a função `contaSe`, analise também os argumentos e como é o seu cabeçalho (arquivo `Conjunto.h`). Complete-a no arquivo `Conjunto.c`.
Exemplo de utilização da função `contaSe`:

```c
bool ehPar(int x) {
    return x % 2 == 0;
}
bool ehImpar(int x) {
    return x % 2 != 0;
}
int main() {
    ...
    Conjunto *c = criaConjuntoValores(5, 1, 2, 3, 4, 6);
    int numParesNoConj = contaSe(c, ehPar); // 3
    int numImparesNoConj = contaSe(c, ehImpar); // 2
    ...
    return 0;
}
```

Exercício 9: Assim como no exercício anterior, sua tarefa é completar a função de alta ordem `transforma`. Tal função recebe um conjunto e uma função de "transformação". O objetivo é aplicar a função passada como parâmetro em cada elemento do conjunto.

Exemplo de utilização da função `transforma`:

```c
bool quadrado(int x) {
    return x*x;
}
int main() {
    ...
    Conjunto *c = criaConjuntoValores(5, 1, 2, 3, 4, 6);
    transforma(c, quadrado);
    imprime(c); // {1, 4, 9, 16, 36}
    ...
    return 0;
}
```

Você se lembra de já ter visto uma função estilo a `transforma` em algum "lugar"?

Bom trabalho e divirta-se!