Estrutura Abstrata de Dados
Árvore Binária de Pesquisa (ABP)

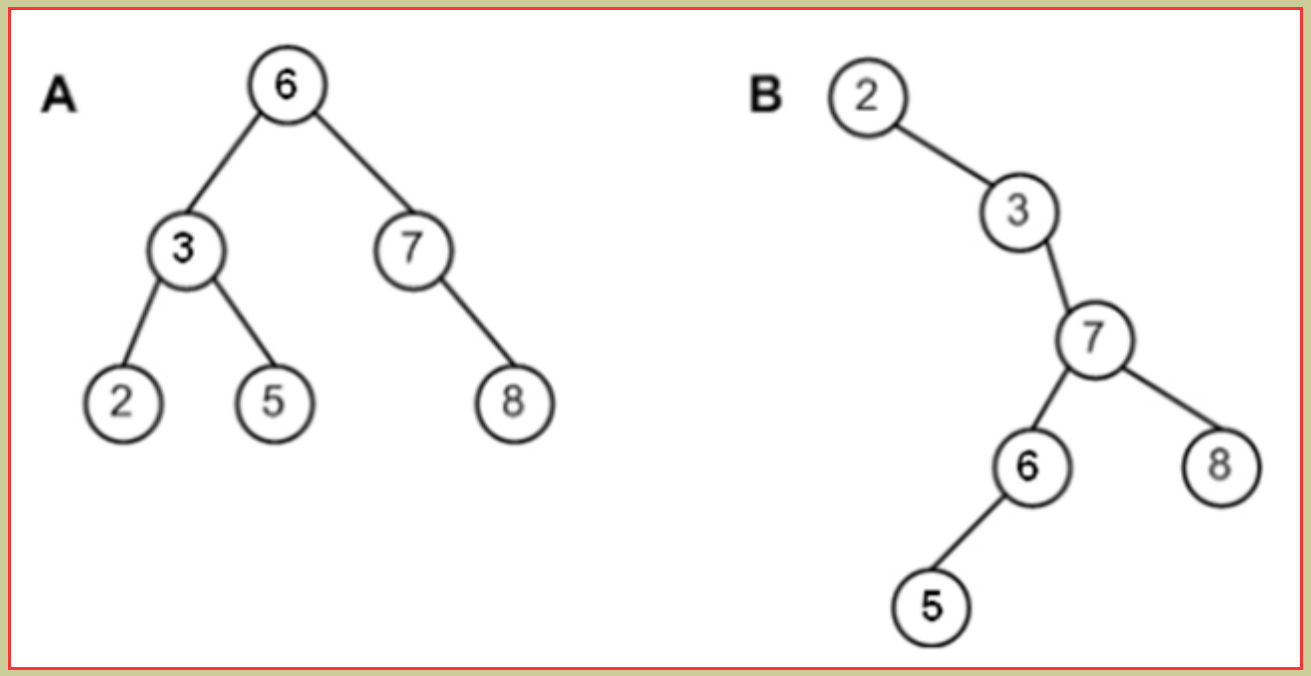
Implementação com ligações simples
(usando ponteiros e memória dinâmica)

Árvore Binária de Pesquisa

Definição

- É uma **árvore binária** em que a chave (elemento) guardada nos seus nodos
  - determina a sua posição de colocação na árvore
  - a chave de um nodo obedece às seguintes regras:
    - é maior do que a chave de qualquer nodo da sua subárvore esquerda
    - é menor do que a chave de qualquer nodo da sua subárvore direita
- Desta definição decorre que não podem existir nodos com chaves iguais
- Desta forma, ao percorrer a árvore *em ordem*, os seus elementos são visitados por *ordem crescente* das suas chaves
- Cada nodo pode ser visto como a raíz de duas árvores
  - a árvore *esquerda* com nodos cujas chaves são inferiores à chave da raíz
  - a árvore *direita* com nodos cujas chaves são superiores à chave da raíz.

Exemplos



Árvore Binária vs. Árvore Binária de Pesquisa

Árvore binária

- As chaves dos elementos de uma AB encontram-se distribuídos pela árvore sem qualquer relação de ordem

Árvore binária de pesquisa

- As chaves dos elementos de uma ABP encontram-se distribuídos pela árvore ordenados por ordem crescente, ao ser percorrida em-ordem
- O objetivo de organizar dados em ABP é facilitar a procura de um elemento
  - a partir da raíz e da chave a ser encontrada, é possível saber qual o caminho a ser percorrido até encontrar o elemento/nodo com aquela chave
  - basta verificar se a chave do elemento procurado é maior ou menor à chave do elemento na posição atual
- Praticamente todas as operações que se aplicam a uma AB também se aplicam a uma ABP

Operações básicas sobre uma ABP

Criar uma ABP

Libertar/destruir uma ABP

Verificar se uma ABP está vazia

Mostrar os elementos de uma ABP: pré-ordem, em-ordem e pós-ordem

Determinar a altura de uma ABP

Pesquisar um elemento numa ABP

Atualizar um elemento de uma ABP

Inserir um elemento numa ABP

Remover um elemento de uma ABP

Nodos e ligações

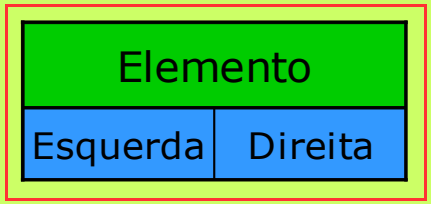
Estrutura

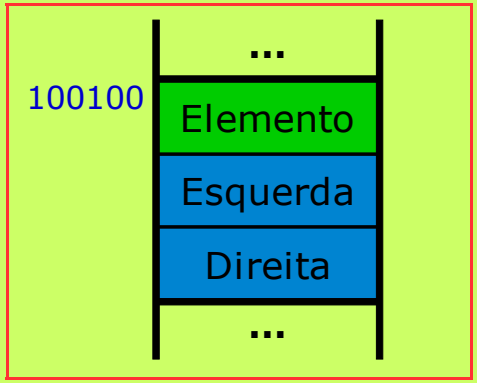
- Como uma ABP é uma AB, cada nodo pode ser representado pela mesma estrutura; ou seja, contendo os seguintes dados
  - a **informação** propriamente dita que se pretende guardar e tratar
  - as **ligações** a outros dois nodos, correspondentes
    - à raiz da **subárvore esquerda**
    - à raiz da **subárvore direita**
- As **ligações** são feitas através de **apontadores**/**ponteiros**, os quais indicam onde
  - está a raiz da subárvore **esquerda**
  - está a raiz da subárvore **direita**

Definição de um nodo

- A definição pode ser a mesma usada para a Árvore Binária, em linuagem C

```
struct  NodoABP {
    INFOABP  Elemento;
    struct NodoABP  *Esquerda;
    struct NodoABP  *Direita;
};
typedef  struct NodoABP  *PNodoABP;
```





- definição implementada através de uma estrutura composta por três campos:
  - **Elemento**: INFOABP é o tipo de dados que contém a informação a guardar e tratar
  - **Esquerda**: ligação à raiz da subárvore **esquerda**
  - **Direita**: ligação à raiz da subárvore **direita**
- definição perante um problema de circularidade, pois os campos **Esquerda** e **Direita**
  - são apontadores para elementos do tipo **NodoABP**
  - fazem parte da estrutura **NodoABP**

Operações sobre a estrutura INFOABP

Mostrar os dados de um elemento do tipo INFOABP

void **mostrarElementoABP** (**INFOABP** X)

- operação que mostra os dados/informação do elemento X do tipo INFOABP

Criar um elemento do tipo INFOABP

INFOABP criarElementoABP ()

- operação para criar um elemento do tipo INFOABP, com dados vindos de fonte adequada

Comparar dois elemento do tipo INFOABP

int **compararElementosABP** (**INFOABP** X, **INFOABP** Y)

- operação que compara dois elementos do tipo INFOABP, segundo um dos seus campos (chave)
- devolve: **-1** (X < Y), **0** (X = Y) ou **1** (X > Y)

Operações básicas sobre um nodo

Criar um nodo de uma ABP

- Adaptar a partir da implementada para Árvores Binárias

```
PNodoABP criarNodoABP (INFOABP X)
```

Libertar um nodo de uma ABP

- Adaptar a partir da implementada para Árvores Binárias

```
PNodoABP libertarNodoABP (PNodoABP P)
```

Operações básicas sobre uma ABP

Criar uma ABP (vazia)

- Adaptar a partir da implementada para Árvores Binárias

```
PNodoABP criarABP ()
```

Libertar/destruir uma ABP

- Adaptar a partir da implementada para Árvores Binárias

```
PNodoABP destruirABP (PNodoABP T)
```

Verificar se uma ABP está vazia

- Adaptar a partir da implementada para Árvores Binárias

```
int ABPVazia (PNodoABP T)
```

Mostrar os elementos de uma ABP

- Adaptar a partir das implementadas para Árvores Binárias

```
void mostrarEmOrdemABP (PNodoABP T)
void mostrarPreOrdemABP (PNodoABP T)
void mostrarPosOrdemABP (PNodoABP T)
```

Determinar a altura de uma ABP

- Adaptar a partir da implementada para Árvores Binárias

```
int alturaABP (PNodoABP T)
```

Pesquisar um elemento numa ABP

Entrada: o elemento X a pesquisar e uma ABP T

Saída: ponteiro para o nodo com o elemento a pesquisar (existe); NULL (não existe)

```
PNodoABP pesquisarABP (INFOABP X, PNodoABP T)
{
  if (T == NULL)
     return NULL;
  if (compararElementosABP(X, T→Elemento) == 0)     //  X = T→Elemento
     return T;
  if (compararElementosABP(X, T→Elemento) == -1)    //  X < T→Elemento
     return pesquisarABP(X, T→Esquerda);
  else
     return pesquisarABP(X, T→Direita);
}
```

Atualizar um elemento de uma ABP

- Como o valor da chave pode também ser atualizado, então esta operação tem que obedecer à definição de ABP

Entrada: a informação atual (chave) X, a nova informação Y e uma ABP T
Saída: ponteiro para a raíz da árvore T (pode ser diferente do inicial)

```
PNodoABP atualizarElementoABP (INFOABP X, INFOABP Y, PNodoABP T)
{
  PNodoABP  P;
  P = pesquisarABP(X, T);
  if (P != NULL) {
     T = removerABP(X, T);
     T = inserirABP(Y, T);
  }
  return T;
}
```

- Nota: se o valor da chave não sofrer atualização, então usar a operação para AB

Inserir um elemento numa ABP

- Este processo tem que obedecer à definição de ABP
  - a chave do elemento a inserir não pode ainda existir na árvore e
  - a sua posição de inserção tem que assegurar que a chave de um elemento é
    - maior do que as chaves de todos os elementos da sua subárvore esquerda e
    - menor do que as chaves de todos os elementos da sua subárvores direita
  - o novo nodo com este elemento é sempre inserido como folha da árvore

Inserir um elemento numa ABP

- Operação só realizada se o elemento a inserir ainda não existir em T

Entrada: o elemento X a inserir e uma ABP T
Saída: a ABP T atualizada, após a inserção de X em T

```
PNodoABP inserirABP (INFOABP X, PNodoABP T)
{
  if (T == NULL) {
     T = criarNodoABP(X);
     return T;
  }
  if (compararElementosABP(X, T→Elemento) == -1)    // X < T→Elemento
     T→Esquerda = inserirABP(X, T→Esquerda);
  else
     T→Direita = inserirABP(X, T→Direita);
  return T;
}
```

Remover um elemento de uma ABP

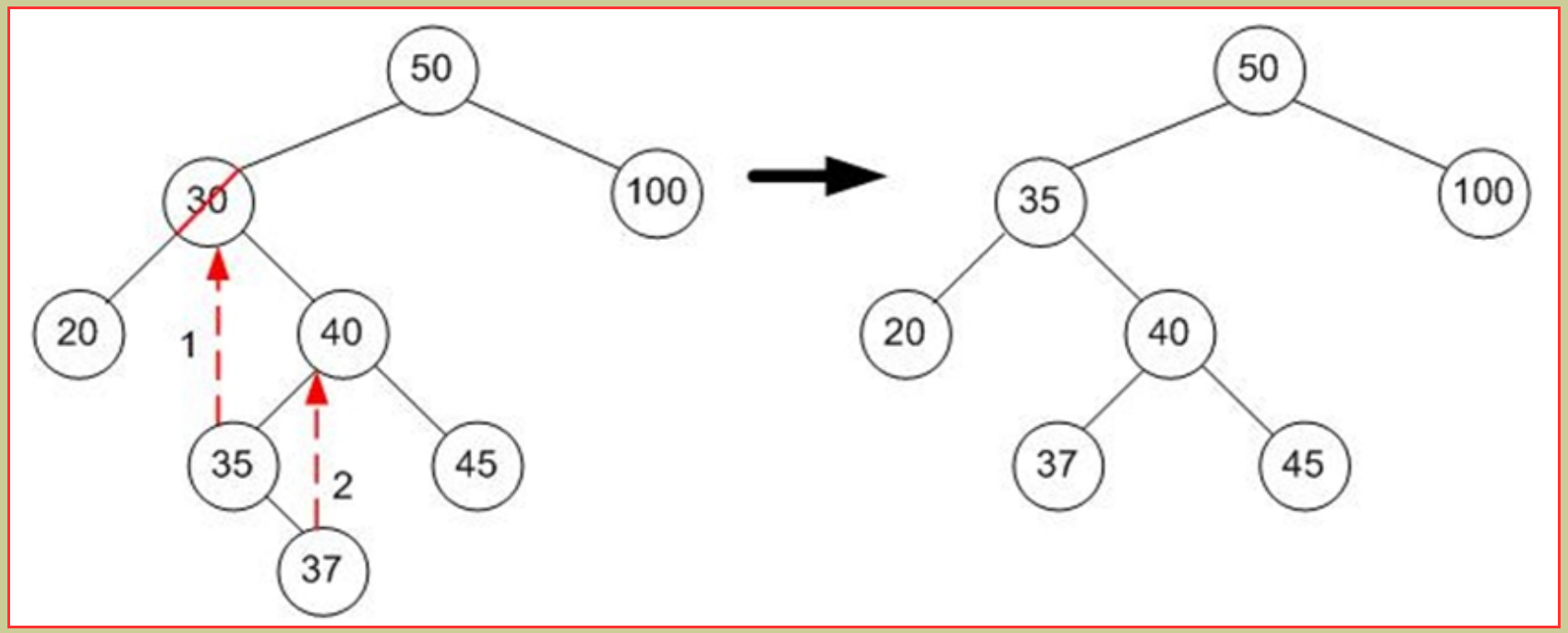
- Considerar três casos distintos, segundo as características do nodo a remover
  - se é um nodo sem filhos (folha)
  - se é um nodo com um único filho
  - se é um nodo com dois filhos
- Mecanismos para tratar os dois primeiros casos são
  - os mesmos descritos para o caso de árvores binárias genéricas
- Mecanismo para tratar o terceiro caso, remoção de um nodo com dois filhos,
  - tem em conta as características de uma ABP (os elementos estão organizados segundo um critério)

Remover um elemento de uma ABP

- Remoção de um nodo com dois filhos
  - O mecanismo de remoção implica substituir este nodo pelo nodo com chave mais "próxima" (maior ou menor) da chave do elemento a ser removido
  - Esta operação pode realizar-se de duas formas diferentes:
    - substituir a informação associada ao nodo a ser removido pela informação do seu sucessor, que é o nodo mais à esquerda da subárvore direita (é o menor elemento da subárvore direita, que é maior do que qualquer elemento da subárvore esquerda)
    - substituir a informação associada ao nodo a ser removido pelo seu antecessor, que é o nodo mais à direita da subárvore esquerda (é o maior elemento da subárvore esquerda, que é menor do que qualquer elemento da subárvore direita)

Remover um elemento de uma ABP

- Remoção de um nodo com dois filhos
  - exemplo: remover o nodo com **30** da árvore em baixo



  - substituir o elemento do nodo com 30 pelo do nodo
    - com **35** (sucessor de 30 na ABP), ou
    - com 20 (antecessor de 30 na ABP)
  - remover a folha com **35** ou com 20, conforme o caso usado

Remover um elemento de uma ABP

- Operação a realizar se o elemento a remover existe em T

Entrada: o elemento X a remover e uma ABP T (X existe em T)
Saída: a ABP T atualizada, após a remoção de X em T

```
PNodoABP removerABP (INFOABP X, PNodoABP T)
{
  if (compararElementosABP(X, T→Elemento) == 0) {       //  X == T→Elemento
     T = removerNodoABP(T);
     return T;
  }
  if (compararElementosABP(X, T→Elemento) == -1)        // X < T→Elemento
     T→Esquerda = removerABP(X, T→Esquerda);
  else
     T→Direita = removerABP(X, T→Direita);
  return T;
}
```

Remover um elemento de uma ABP – operação auxiliar

Entrada: ponteiro para o nodo a remover (raíz de subárvore não vazia)
Saída: ponteiro para a raíz da subárvore após remoção do elemento

```
PNodoABP removerNodoABP (PNodoABP T) {
  PNodoABP  P;
  INFOABP  X;
  if (T→Esquerda == NULL && T→Direita == NULL) {      // T é uma folha
     T = libertarNodoABP(T);
     return T;
  }
  if (T→Esquerda == NULL) {      // T só tem um filho: o direito
     P = T;
     T = T→Direita;
     P = libertarNodoABP(P);
     return T;
  }
...
```

Remover um elemento de uma ABP – operação auxiliar

```
...
  if (T→Direita == NULL) {          // T só tem um filho: o esquerdo
      P = T;
      T = T→Esquerda;
      P = libertarNodoABP(P);
      return T;
  }
  // T tem 2 filhos: remover o nodo sucessor/antecessor e copiar a sua informação
  if (alturaABP( T→Esquerda) < alturaABP( T→Direita)
      T→Direita = substituirNodoDireita(T→Direita, &X);          // 1º caso
  else
      T→Esquerda = substituirNodoEsquerda(T→Esquerda, &X);     // 2º caso
  // substituir o elemento do nodo apontado por T pelo elemento da folha substituta
  T→Elemento = X;
  return T;
}
```

Remover um elemento de uma ABP – operação auxiliar

- Operação para substituir a informação associada ao nodo a ser removido pela informação do seu **sucessor** (o nodo mais à esquerda da subárvore direita)

Entrada: ponteiro para o nodo que se pretende excluir (raíz de subárvore não vazia)
Saída: ponteiro para a raíz da árvore e o elemento do nodo removido

```
PNodoABP substituirNodoDireita (PNodoABP T, INFOABP *X) {
  PNodoABP  P;
  if (T→Esquerda == NULL) {
     *X = T→Elemento;
     P = T;
     T = T→Direita;
     P = libertarNodoABP(P);
     return T;
  }
  T→Esquerda = substituirNodoDireita(T→Esquerda, X);
  return T;
}
```

Remover um elemento de uma ABP – operação auxiliar

- Operação para substituir a informação associada ao nodo a ser removido pela informação do seu **antecessor** (o nodo mais à direita da subárvore esquerda)

Entrada: ponteiro para o nodo a remover (raíz de subárvore não vazia)
Saída: ponteiro para a raíz da árvore e o elemento do nodo removido

```
PNodoABP substituirNodoEsquerda (PNodoABP T, INFOABP *X) {
  PNodoABP P;
  if (T→Direita == NULL) {
     *X = T→Elemento;
     P = T;
     T = T→Esquerda;
     P = libertarNodoABP(P);
     return T;
  }
  T→Direita = substituirNodoEsquerda(T→Direita, X);
  return T;
}
```