ALGORITMOS Y ESTRUCTURA DE DATOS

Biblioteca de Templates

Operaciones sobre arrays

Función: `agregar`

Agrega el valor `v` **al final del *array*** `arr` **e incrementa su longitud** `len`.

```
template <typename T> void agregar(T arr[], int& len, T v);
```

Función: `buscar`

Busca la primer ocurrencia de `v` **en** `arr`; **retorna su posición o un valor negativo si** `arr` **no contiene a** `v`. **La función** `criterio` **debe recibir dos parámetros *t* y *k* de tipo** `T` **y** `K` **respectivamente y retornar un valor negativo, cero o positivo según *t* sea menor, igual o mayor que *k*.**

```
template <typename T, typename K>
int buscar(T arr[], int len, K v, int (*criterio)(T,K));
```

Función: `eliminar`

Elimina el valor ubicado en la posición `pos` **del *array*** `arr`, **decrementando su longitud** `len`.

```
template <typename T> void eliminar(T arr[], int& len, int pos);
```

Función: `insertar`

Inserta el valor `v` **en la posición** `pos` **del *array*** `arr`, **incrementando su longitud** `len`.

```
template <typename T> void insertar(T arr[], int& len, T v, int pos);
```

Función: `insertarOrdenado`

Inserta el valor `v` **en el *array*** `arr` **en la posición que corresponda según el criterio de precedencia que establece la función** `criterio`; **esta función recibe dos parámetros *v1*, y *v2* ambos de tipo** `T` **y retorna un valor negativo, cero o positivo según *v1* sea menor, igual o mayor que *v2* respectivamente.**

```
template <typename T>
int insertaOrdenado(T arr[], int& len, T v, int (*criterio)(T,T));
```

Función: buscarEInsertarOrdenado

Busca el valor v en el *array* arr; si lo encuentra entonces retorna su posición y asigna true al parámetro enc. De lo contrario lo inserta donde corresponda según el criterio criterio, asigna false al parámetro enc y retorna la posición en donde finalmente quedó ubicado el nuevo valor.

```
template <typename T>
int buscarEInsertarOrdenado(T arr[],int& len,T v,bool& enc,int (*criterio)(T,T));
```

Función: ordenar

Ordena el *array* arr según el criterio de precedencia que establece la función criterio.

```
template <typename T> void ordenar(T arr[], int len, int (*criterio)(T,T));
```

Función: busquedaBinaria

Busca el elemento v en el *array* arr que debe estar ordenado según el criterio criterio. Retorna la posición en donde se encontró el elemento (si se encontró) o la posición en donde dicho elemento podría ser insertado manteniendo el criterio que establece la función criterio que recibe cómo parámetro.

```
template<typename T, typename K>
int busquedaBinaria(T a[], int len, K v, int (*criterio)(T, K), bool& enc);
```

Operaciones sobre estructuras dinámicas

Nodo

```
template <typename T>
struct Nodo
{
   T info;       // valor que contiene el nodo
   Nodo<T>* sig; // puntero al siguiente nodo
};
```

Operaciones sobre Listas

Función: agregar

Agrega un nodo con valor v al final de la lista direccionada por p.

```
template <typename T> void agregar(Nodo<T>*& p, T v);
```

Función: liberar

Libera la memoria que insumen todos los nodos de la lista direccionada por p; finalmente asigna NULL a p.

```
template <typename T> void liberar(Nodo<T>*& p);
```

Función: buscar

Retorna un puntero al primer nodo de la lista direccionada por p cuyo valor coincida con v, o NULL si ninguno de los nodos contiene a dicho valor. La función criterio debe comparar dos elementos *t* y *k* de tipo T y K respectivamente y retornar un valor: menor, igual o mayor que cero según: *t<k, t=k o t>k*.

```
template <typename T, typename K>
Nodo<T>* buscar(Nodo<T>* p, K v, int (*criterio)(T,K));
```

Función: `eliminar`

Elimina el primer nodo de la lista direccionada por `p` cuyo valor coincida con `v`. La función asume que existe al menos un nodo con el valor que se desea eliminar.

```
template <typename T, typename K>
void eliminar(Nodo<T>*& p, K v, int (*criterio)(T,K));
```

Función: `eliminarPrimerNodo`

Elimina el primer nodo de la lista direccionada por `p` y returna su valor. Si la lista contiene un único nodo entonces luego de eliminarlo asignará `NULL` a `p`.

```
template <typename T> T eliminarPrimerNodo(Nodo<T>*& p);
```

Función: `insertarPrimerNodo`

Crea un nodo cuyo valor será `v` y lo inserta como primer elemento de la lista. Luego de invocar a esta función `p` tendrá la dirección del nuevo nodo.

```
template <typename T> T insertarPrimerNodo(Nodo<T>*& p, T v);
```

Función: `insertarOrdenado`

Inserta un nodo en la lista direccionada por `p` respetando el criterio de ordenamiento que establece la función `criterio`.

```
template <typename T>
Nodo<T>* insertarOrdenado(Nodo<T>*& p, T v, int (*criterio)(T,T));
```

Función: `ordenar`

Ordena la lista direccionada por `p`; el criterio de ordenamiento será el que establece la función `criterio`.

```
template <typename T> void ordenar(Nodo<T>*& p, int (*criterio)(T,T));
```

Función: `buscarEInsertarOrdenado`

Busca en la lista direccionada por `p` la ocurrencia del primer nodo cuyo valor sea `v`. Si existe dicho nodo entonces retorna su dirección; de lo contrario lo inserta respetando el criterio de ordenamiento establecido por la función `criterio` y retorna la dirección del nodo insertado. Finalmente asigna `true` o `false` al parámetro `enc` según el valor `v` haya sido encontrado o insertado.

```
template <typename T>
Nodo<T>* buscarEInsertarOrdenado(Nodo<T>*& p,T v,bool& enc,int (*criterio)(T,T));
```

Operaciones sobre pilas

Función: `apilar` (***push***)

Apila el valor `v` a la pila direccionada por `p`.

```
template <typename T> void apilar(Nodo<T>*& p, T v);
```

Función: desapilar (*pop*)

Remueve y retorna el valor que se encuentra en la cima de la pila direccionada por p.

```
template <typename T> T desapilar(Nodo<T>*& p);
```

Opeaciones sobre colas (implementación: lista enlazada con dos punteros)

Función encolar

Encola el valor v en la cola direccionada por p y q implementada sobre enlazada.

```
template <typename T> void encolar(Nodo<T>*& p, Nodo<T>*& q, T v);
```

Función desencolar

Desencola y retorna el próximo valor de la cola direccionada por p y q implementada sobre una lista enlazada.

```
template <typename T> T desencolar(Nodo<T>*& p, Nodo<T>*& p);
```

Operaciones sobre archivos

Función read

Lee un registro de tipo T desde el archivo f. Para determinar si llegó o no el *eof* se debe utilizar la función feof.

```
template <typename T> T read(FILE* f);
```

Función write

Escribe el contenido del registro v, de tipo T, en el archivo f.

```
template <typename T> void write(FILE* f, T v);
```

Función seek

Mueve el indicador de posición del archivo f hacia el registro número n.

```
template <typename T> void seek(FILE* f, int n);
```

Función fileSize

Retorna la cantidad de registros que tiene el archivo f.

```
template <typename T> long fileSize(FILE* f);
```

Función filePos

Retorna el número de registro que está siendo apuntado por el indicador de posición del archivo.

```
template <typename T> long filePos(FILE* f);
```

Función busquedaBinaria

Busca el valor v en el archivo f; retorna la posición del registro que lo contiene o -1 si no se encuentra el valor.

```
template <typename T, typename K>
int busquedaBinaria(FILE* f, K v, int (*criterio)(T,K));
```