1.2 – `ConjuntoChar` con elementos ordenados

Estructuras de Datos
Facultad de Informática - UCM

Retomemos la definición de `ConjuntoChar` que utiliza un array con los caracteres contenidos en el conjunto:

```
class ConjuntoChar {
public:
  ConjuntoChar();

  bool pertenece(char letra) const;
  void anyadir(char letra);

private:
  int num_elems;
  char elems[MAX_CHARS];
};
```

Si `c` es una instancia de `ConjuntoChar`, suponemos la siguiente función de abstracción e invariante de representación:

$$f(\texttt{c}) = \{\texttt{c.elems}[i] \mid 0 \leq i < \texttt{c.num\_elems}\}$$

$$\begin{aligned} I(\texttt{c}) = \quad & 0 \leq \texttt{c.num\_elems} \leq \texttt{MAX\_CHARS} \\ & \wedge\, \forall i : 0 \leq i < \texttt{c.num\_elems} \Rightarrow \texttt{c.elems}[i] \in \{A..Z\} \\ & \wedge\, \forall i, j : 0 \leq i < j < \texttt{c.num\_elems} \Rightarrow \texttt{c.elems}[i] < \texttt{c.elems}[j] \end{aligned}$$

El invariante indica que los caracteres del conjunto deben estar ordenados en el array `elems`.

1. Implementa el método `anyadir` para que inserte el elemento dado en el array `elems` de manera ordenada, preservando así el invariante. Si el elemento ya existía en el array, no hace nada.
2. Implementa el método `pertenece` para que realice una búsqueda eficiente del elemento pasado como parámetro. Puedes utilizar la función `binary_search` definida en el fichero de cabecera `<algorithm>`.
3. Indica el coste en tiempo de ambas operaciones.

Solución

```
class ConjuntoChar {
public:
  ConjuntoChar();

  bool pertenece(char letra) const;
  void anyadir(char letra);

private:
  int num_elems;
  char elems[MAX_CHARS];

  friend std::ostream & operator<<(std::ostream &out, const ConjuntoChar &c);
};

ConjuntoChar::ConjuntoChar() {
  num_elems = 0;
}

// O(log num_elems)
bool ConjuntoChar::pertenece(char l) const {
  return std::binary_search(elems, elems + num_elems, l);
}



// O(num_elems)
void ConjuntoChar::anyadir(char l) {
  int i = 0;
  while (i < num_elems && l > elems[i]) {
    i++;
  }

  if (l != elems[i]) {
    assert (num_elems != MAX_CHARS);
    for (int j = num_elems; j > i; j--) {
      elems[j] = elems[j - 1];
    }
    num_elems++;
    elems[i] = l;
  }
}
```