Úkolem je vytvořit šablonu třídy, který bude implementovat generický index.

Implementovaná třída `CIndex` dostane při vytváření parametrem kolekci s posloupností prvků. Tuto posloupnost prvků bude indexovat. Posloupnost může být:

- `string` - řetězec znaků (C++ string prvků typu char),
- `vector<T>` - vektorem hodnot (nějakého typu T),
- `list<T>` - seznamem hodnot (nějakého typu T).

Nad takto zaindexovanou posloupností chceme vyhledávat. Tedy zadáme nějakou posloupnost prvků (stejného typu) a chceme zjistit, zda ji indexovaná posloupnost někde obsahuje. Výsledkem bude množina pozic, kde je hledaná posloupnost nalezena.

Aby byla indexovací třída obecnější, je vylepšena ještě možností zadat vlastní porovnávač (nepovinný druhý generický parametr). Vyhledávání tedy nemusí probíhat na přesnou shodu, pomocí vlastního porovnávače můžeme zadat, že při porovnávání např. nerozlišujeme malá a velká písmena. Porovnávač bude mít podobu jakou mají porovnávače v STL: bude se jednat o funkci, funktor nebo C++11 lambda výraz, který pro dvojici prvků typu T v posloupnosti rozhodne, který je v požadovaném porovnání menší. Pokud porovnávač není zadán, použije pro porovnávání prvků operátor < platný pro ukládané hodnoty prvků.

Vlastní prvky v posloupnostech (prvky typu T) mohou být libovolné. Příkladem je znak (char), celé číslo (int) nebo řetězec (string). Obecně víte, že pro typ T jsou definované následující operace:

- kopírování (operátorem = a kopírujícím konstruktorem),
- porovnávání "menší než" operátorem < nebo dodaným komparátorem,
- uvolňování (destruktor),
- další operace mohou, ale nemusí být s typem T dostupné, tedy Vaše implementace se na ně obecně nemůže spolehnout. Pozor, nemusí být k dispozici implicitní konstruktor, operátor ==, operátor !=, ...

Odevzdávejte soubor, který obsahuje implementovanou šablonu třídy `CIndex` a další Vaše podpůrné třídy. Třída musí splňovat veřejné rozhraní podle ukázky - pokud Vámi odevzdané řešení nebude obsahovat popsané rozhraní, dojde k chybě při kompilaci. Do třídy si ale můžete doplnit další metody (veřejné nebo i privátní) a členské proměnné. Odevzdávaný soubor musí obsahovat jak deklaraci třídy (popis rozhraní) tak i definice metod, konstruktoru a destruktoru. Je jedno, zda jsou metody implementované inline nebo odděleně. Odevzdávaný soubor nesmí obsahovat vkládání hlavičkových souborů a funkci `main`. Funkce `main` a vkládání hlavičkových souborů může zůstat, ale pouze obalené direktivami podmíněného překladu jako v ukázce níže.

Při řešení úlohy využijte STL. Můžete využívat většinu konstrukcí do C++ 17 včetně.

Úloha obsahuje povinné a bonusové testy. Základním testem projde řešení hledající výskyty naivním algoritmem. Pro získání bonusů je potřeba implementovat algoritmy efektivnější. Při návrhu můžete předpokládat:

- Jednou zaindexovaná posloupnost je prohledávaná vícekrát. Je tedy rozumné věnovat čas předzpracování, kterým se zkrátí vlastní vyhledávání. Můžete počítat s tím, že metoda `search` je volaná v průměru 100x na jednu instanci.
- Obecná implementace s generickým typem prvku a s obecným komparátorem se testuje v povinných testech. Testy rychlosti pracují s datovým typem `string` a používají implicitní komparátor. Částečná specializace šablony třídy pro takto omezené vstupy může výrazně zrychlit výpočet.
- Pro zrychlení vyhledávání se hodí algoritmus KMP, pro indexaci se dá využít suffixové pole nebo DAWG.