POS1 – Übungen zu dynamischen Listen in C++ **Schuljahr:** 2023/24

Erstellen Sie ein cpp-Programm, welches eine sortierte, dynamische Liste von Strings implementiert!

Folgende Funktionalitäten bzw. Merkmale sollen vorhanden sein:

a) sortiertes Einfügen (iterativ und rekursiv)
b) aufsteigende und absteigende Ausgabe (iterativ [nur aufsteigend] und rekursiv)
c) Suchen nach einem Element (iterativ und rekursiv)
d) Löschen eines Elementes aus der Liste (iterativ und rekursiv)
e) Ermitteln der Größe (Länge) der Liste (iterativ und rekursiv)
f) Auflösen der Liste und Freigabe des Speichers (iterativ und rekursiv)
g) Bedienung des Programms über Menü oder Kommandozeilenargumente
h) Aufteilung des Codes in mehrere Module (Hauptprogramm, Menü, Liste)

Implementierung:

a) Die Funktionen `void insert_iterative(node** r, string value)` und `void insert_recursive(node** r, string value)` sind als Musterlösung in der Vorgabe bereits enthalten. Die Funktionen sollen mittels Iteration bzw. Rekursion die richtige Stelle zum Einfügen in der sortierten Liste finden und das neue Element an der entsprechenden Stelle einfügen!
b) Die Funktionen `void print_iterative(node* r)`, `void print_recursive(node* r)` und `void print_recursive_reverse(node* r)` (inkl. der notwendigen Hilfsfunktionen) zum iterativen bzw. rekursiven Ausgeben der Liste in aufsteigender bzw. absteigender Reihenfolge sind als Musterlösung in der Vorgabe bereits enthalten. Beachten Sie, dass die Ausgabe in absteigender Reihenfolge nur mittels Rekursion möglich ist!
c) Erstellen Sie die Funktionen `bool find_iterative(node* r, string value)` und `bool find_recursive(node* r, string value)` zum Suchen eines in der Liste. Die Suche nach dem entsprechenden Element ist dabei iterativ bzw. rekursiv zu ermitteln. Die Funktionen geben jeweils `true` oder `false` als Funktionsergebnis zurück, je nachdem, ob der Wert gefunden wurde oder nicht.
d) Erstellen Sie die Funktionen `bool remove_iterative(node** r, string value)` und `bool remove_recursive(node** r, string value)` zum Entfernen eines Elementes aus der sortierten Liste. Die Suche nach dem entsprechenden Element ist dabei iterativ bzw. rekursiv zu ermitteln. Die Funktionen geben jeweils `true` oder `false` als Funktionsergebnis zurück, je nachdem, ob der Wert gefunden wurde oder nicht.
e) Erstellen Sie die Funktionen `int getsize_iterative(node* r)` und `int getsize_recursive(node* r)` zum Ermitteln der Anzahl der in der Liste enthaltenen Elemente. Die Ermittlung der Anzahl der enthaltenen Elemente ist dabei iterativ bzw. rekursiv zu ermitteln und als Funktionsergebnis an die aufrufende Stelle zurückzugeben.
f) Erstellen Sie die Funktionen `void delete_iterative(node** r)` und `void delete_recursive(node** r)` zum Löschen der gesamten Liste inkl. der Freigabe des Speichers. Das jeweils gerade gelöschte Element soll an der Konsole ausgegeben werden. Beachten Sie, dass die rekursive Variante die Liste vom Ende her auflösen soll!

g) Wird das Programm ohne Kommandozeilenparameter aufgerufen, soll der weitere Programmablauf mithilfe eines Menüs gesteuert werden können. Das Menü muss dabei sämtliche obig angeführten Funktionalitäten beinhalten (siehe Demo-Programm `soli_solution`)
Ansonsten sollen sämtliche Parameter in die Liste sortiert eingefügt werden, aufsteigend und absteigend ausgegeben und danach das Programm beendet werden.

h) Verwenden Sie zur Lösung der Aufgabe die Dateien `menu.h`, `menu.cpp`, `sortedlist.h`, `sortedist.cpp`, `soli.h` und `soli.cpp` auf (in der Vorgabe enthalten)! Sämtliche leere Funktionen sind zu implementieren!

Hinweise:

- Halten Sie sich an die üblichen Programmierrichtlinien (Kommentare, Header, Einrückungen, …)!
- Verwenden Sie zur Lösung der Aufgabenstellung die vorhandenen Dateien und ergänzen Sie die leeren Funktionen!
- Halten Sie sich möglichst genau an die Angabe und das vorgebebene Demo-Programm `soli_solution`!
- Beachten Sie, dass sämtlicher mit `new` angeforderter Speicher vor Beendigung des Programms wieder freigegeben werden muss!