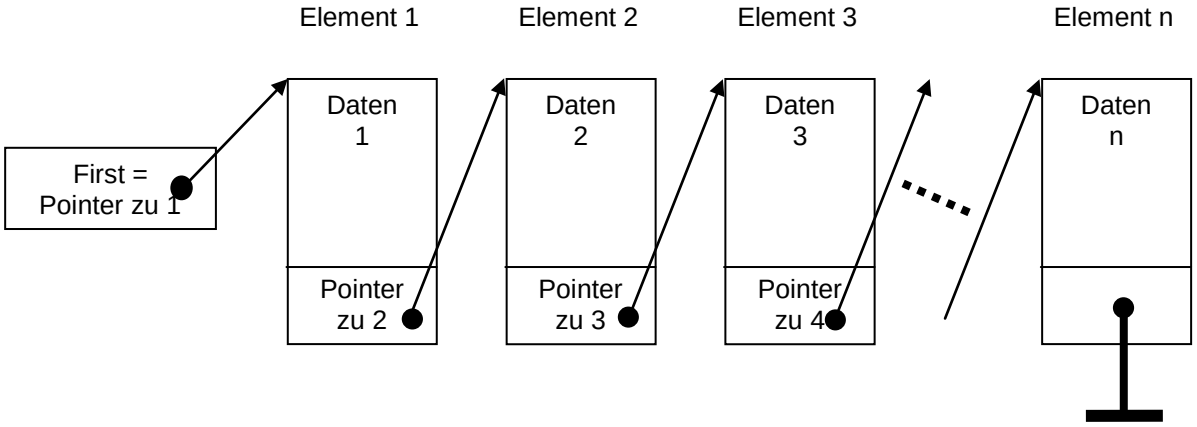
Praktikum 2: Objektorientierte Programmierung einer verketteten Liste

Das Ziel dieses Praktikums ist die Erstellung eines Programms zur Bearbeitung der Einträge in einer verketteten Liste von Strings bei gleichzeitiger Verwaltung einer Warteschlange (FIFO). In einem Hauptprogramm soll die Funktionsfähigkeit demonstriert werden.

Eine verkettete Liste ist eine Folge von Datenelementen, bei denen zusätzlich zu den zu verwaltenden Daten ein Pointer auf das nächste Element angelegt wird. Hierbei ist First ein Pointer auf das erste Element (Anker). Bei jedem Listenelement wird außer den Daten ein Zeiger auf das Nachfolge-Element angelegt, das letzte Element enthält als Adresse einen vorgegebenen Wert, der nicht als gültige Adresse interpretiert werden kann (= NULL/nullptr).



Erstellen Sie die Klasse EVKD zur Verwaltung von ASCII-Zeichen in Form einer Zeichenkette. Objekte von EVKD sollen als Elemente einer einfachverketteten Liste verwendet werden können.

EVKD habe als private Attribute einen Pointer (Variablenname Daten) auf ein Array von char-Variablen (kein String, also nicht durch '\0' abgeschlossen!), eine int-Variable für die Anzahl der Zeichen in Daten (Variablenname AnzChar) und eine Variable des Typs EVKD * mit dem Namen Next.

Erstellen Sie für die Objekterzeugung einen Konstruktor mit der Signatur
EVKD(char * LPSZDaten, EVKD * Next).
LPSZDaten ist ein Pointer für ein char-Array, dessen letztes Element das Stringendezeichen ('\0') enthält. Next soll bei Verwendung von EVKD-Objekten in einer verketteten Liste das Nachfolgeelement adressieren.

Geben Sie die get-Methoden der Attribute an. Die get-Methode für das Attribut Daten mit der Signatur char * getDaten(); gibt ein mit '\0' terminiertes char-Array zurück. Beachten Sie, dass kein Pointer auf gekapselte Daten nach außen gegeben werden darf (außer bei getNext()!)!
Geben Sie zusätzlich für das Attribut Next die set-Methode mit der Signatur:
void setNext(EVKD * Next); an.

Überladen Sie die Operatoren > und == für Objekte der Klasse EVKD. Die Signaturen dazu sind: bool operator > (EVKD & RHS); und bool operator == (EVKD & RHS);
Hinweis: Erzeugen Sie für den Vergleich Strings (char-Array mit '\0'!) der Inhalte der zu vergleichenden Objekte und verwenden Sie hier die Standard-C-Methode mit der Signatur int strcmp(char * S1, char * S2).

Legen Sie die Klasse QUEUE an für eine einfach verkettete Liste von Objekten der Klasse EVKD mit der Aufgabe eines FIFO-Speichers. QUEUE hat die privaten Attribute
EVKD * Enter; als Pointer auf den Beginn der Liste (Einfügestelle des FIFO),
EVKD * Last; als Pointer auf das Ende der Liste (Entnahmestelle des FIFO) und
long AnzElem; für die Anzahl der tatsächlichen Einträge der Liste.
Im Standardkonstruktor von QUEUE werden diese Attribute mit den Konstanten NULL/nullptr (für die Pointer) und 0 initialisiert, das heißt, zu Beginn ist die Warteschlange leer.
Verwenden Sie hierfür die Initialisierungsliste!

Erstellen Sie die Methoden void queueIn (char * InText), welche dem FIFO ein neues EVKD-Objekt mit InText als Daten hinzufügt und char * queueOut (), welche aus dem FIFO das letzte Element (bezüglich der Warteschlange das Älteste!) entfernt und die Daten des Elementes zurückgibt! Sowohl InText als auch der Rückgabewert von queueOut sind Pointer für char-Arrays, deren letztes Element je das Stringendezeichen ('\0') enthält.

Für die weitere Betrachtung ist ein Sortiermechanismus in der Warteschlange zu programmieren, mit dessen Hilfe die Elemente innerhalb der Warteschlange alphabetisch sortiert werden können. Dazu soll das Selection-Sort Verfahren angewendet werden.

Erstellen Sie hierfür die Methode void einfuegeBei (EVKD * In, int Pos), die das Objekt In vor der durch Pos bezeichneten Stelle in der Liste einfügt und die Methode
EVKD * loesche(int Pos) zum Entfernen des durch Pos angegebenen Elementes. Das erste Element hat Pos = 1, das letzte Pos = AnzElem!

Verwenden Sie diese beiden Methoden in der Methode void selSort() um ein modifiziertes Selection-Sort-Verfahren zu programmieren! Mit Hilfe von selSort sind die Elemente der verketteten Liste der Größe nach (Verwendung des operator >(…) für EVKD-Objekte!!!) zu sortieren, so dass das größte Element als nächstes durch queueOut aus dem FIFO entfernt wird.
Algorithmuskurzfassung des modifizierten Selection-Sort: Finde nach einander, beginnend bei dem ersten bis zum letzten, das jeweils größte Element, nimm es aus der Liste heraus in einen Merker, füge es am Ende der noch nicht sortierten Liste wieder ein (=löschen und einfügen…)

Erstellen Sie die Methode void zeigDich() um den Inhalt der verketteten Liste auf dem Bildschirm inklusive der Adressen zur Kontrolle der Verlinkung (Pointer) auszugeben.

Erstellen Sie in einem Hauptprogramm (main) ein Menü mit den folgenden Einträgen:

1.) Warteschlange erstellen,
2.) Verkettete Liste anzeigen,
3.) Neues Element zu Warteschlange hinzufügen,
4.) Element aus der Warteschlange entfernen
5.) Liste sortieren und
12.) Ende!

Verwenden Sie beim Menüpunkt 1.) Warteschlange erstellen das folgende Array von Strings um ein QUEUE-Objekt zu erstellen.

```
char * Name[] = {"Schmitz, Josef","Mueller, Josi","Schmitz, Anna",
                 "Mueller, Josef","Schmitz, Josi","Mueller, Anna",
                 "Meier, Josef","Zacher, Josi","Anker, Anna"};
```

Die Ausgabe bei 2.) Verkettete Liste anzeigen zeigt in einer Tabelle neben den Daten die zur Kontrolle der verketteten Liste notwendigen Pointer und Elementadressen.

Bei 3.) Neues Element an Warteschlange anhaengen können vom Benutzer Name und Vorname eingegeben werden. Mit diesen Informationen ist ein neues Element der Warteschlange hinzuzufügen.

Durch 4.) Element aus der Warteschlange entfernen wird das Element am Beginn der Warteschlange (das älteste…) aus dieser entfernt und die Daten des Elementes angezeigt.

Menupunkt 5.) Liste sortieren ändert die Reihenfolge in der Warteschlange, so dass das alphabetisch größte Element als nächstes aus der Warteschlange entfernt werden kann.

Erstellen Sie ein Code::Blocks-Projekt für die Version 20.03 mit dem Compiler gcc8.1.0 für Windows, unicode, 32 Bit. Legen Sie für die Klassen EVKD und QUEUE jeweils eine Header und Code-Datei an, die Sie in das Projekt integrieren. In einer Datei mit dem Namen main.cpp befindet sich die Funktion main mit dem Menu.

Weisen Sie die Funktionsfähigkeit mit einer Folge von Screen Shots der Ausführung Ihres Programms nach. Verwenden Sie hierfür bei Menu-Punkt 3.) Ihren Namen und den Namen Ihres Partners, sortieren anschließend die Liste. Demonstrieren Sie die Validität der verketteten Liste durch geeignete Ausgabe der Liste durch 2.) (Screen Shots!).

Abgabe: 03. 07. 2023