RLE и Base64 кодиране

При конструиране на RLE списък програмата създава нов низ и масив със стойности с първоначална дължина 10, като чете низа и брои колко пъти всеки повтарящ се символ се среща. Ако срещне различен символ от този който чете програмата записва предишния символ и съответния брой срещания. При извършване на операциите програмата проверява на коя позиция да се извърши (спрямо разкодирания низ) и не допуска два поредни символа в кодирания списък да са едни и същи.  
При Base64 програмата разбива низа който кодира на битове и го записва в масив от тип bool. Получения се разделя на групи по 6 бита които се интерпретират като един символ от Base64 кодирането и се записват в новия кодиран низ. Ако не достигат битове, така че масивът от битове да се раздели точно на групи от 6, програмата запълва липсващите места с нули и добавя съответно специалният символ ‘=’ в кодирания низ. При извършване на операциите програмата осигурява върнатите кодирани низове, да са подмасиви на първоначалния разкодиран низ.

1. **Class RLЕ**  
   1.1 **член данни:**   
    **char\* dataValue** – всички символи които се срещат в първоначалния низ  
    unsigned\* count – броя на срещания на всеки символ в data Value, като всяко dataValue[i] отговаря на count[i].  
   1.2 конструктори: конструктор по подразбиране, конструктор по низ, конструктор по dataValue и count, копиконструктор.  
   1.3 функции :  
    1.3.1 **оператор за присвояване, оператор за извеждане** - стандартни  
    1.3.2 **decode()** – декодира RLE списъка връща тип char\*  
    1.3.3 **оператори конкатенация + и +=** - създават нови масиви tempDataValue и tempCount, копират последователно двата списъка; операцията + връща нов RLE списък, += променя this.  
    1.3.4 **оператори ++ и -- (в постфиксен и префиксен вариант)** – променят RLE списък като изваждат или добавят единица към count на последния елемент.  
    1.3.5 **оператори(unsigned position, unsigned length)** – връща подсписък на оригиналния, с позиция на декодирания масив и с дължина от length.  
    1.3.6 **оператор(unsigned position, RLE& B)** – вмъква B на позиция position от декодирания масив. Проверява дали първия и последния символ на B съвпадат околните и ги слива.  
    1.3.**7 оператори (==, !=, <, > , <= , <=)** – проверяват дали даден списък е подсписък на друг.  
    1.3.8 **оператор[]** – намира символ на съответната позиция от декодирания масив.  
    1.3.9 **оператор\*** - връща нов списък който съдържа всички уникални символи от оригиналния и съответния брой срещания.  
    1.3.10 **resize -**  ползва се само при конструктора, тъй като не знаем колко ще е голям dataValues без много проверки, dataValue първоначално е с размер 10 и се разширява ако свърши мястото.  
    1.3.11 **селектори getValues и getCounts**.
2. **Class Base64**  
   2.1 **член данни:**  
    **char\* encodedString** – кодираният низ.  
   2.2 **конструктори**  
    конструктор по подразбиране, копиконструктор и конструктор по даден низ – разбива низа на насив от bool\* допълва го, ако не се дели на 3, и след това го разделя на групи от 6 които представят един символ от base64 кодирането.  
   2.3 **функции** оператор за присвояване – стандартен.  
    2.3.1 **decode()** – отново разбива кодирания масив, изважда допълнените нули и го разделя на групи от 8.  
    2.3.2 **оператор[]** – намира символ от декодирания масив.  
    2.3.**3 оператор+ и +=** - конкатенират два низа, като резултата е кодиран низ на конкатенацията на двата разкодирани.  
    2.3.**4 оператор (unsigned startPosition, unsigned length)** –връща кодиран подмасив с дължина length на разкодирания първоначален низ, като почне от startPosition.  
    2.3.5 **оператор (unsigned startPosition, Base64& B)** – вмъква кодирания низ B на позиция startPosition от разкодирания низ на this. Новия кодиран низ, след разкодиране, е разкодирания низ на this с кодирания низ на B на позиция startPosition.  
    2.3.6 **оператори за сравнение (==, !=, >, <, <=, >=)** – проверяват дали кодиран масив е еднакъв, еднакъв или подмасив, или строг подмасив на даден.  
    2.3.7 **deleteCharacters(unsigned startPosition, unsigned length)** – изтрива разкодиран низ от позиция startPosition напред с дължина length. Функцията разбива кодирания низ на битове, и записва в нов масив, като пропуска битовете от startPosition\*8 до   
   startPosition\* 8 + length\*8.  
    2.3.8 **convertEncodedStringToBits(char\*)** – превръща кодиран низ в масив от bool  
   2.4 **помощни функции**:  
    2.4.1 **convertStringToBits(unsigned char\* text)** – разбива низ в масив от bool и го връща.  
    2.4.2 **convertBitsToString(bool\* stringBits, unsigned size)** – конвертира масив от bool в низ (не кодиран).  
    2.4.3 **convertChar(char n )** – връща номера на съответен кодиран символ.  
    2.4.4 **convertInt(int n)** – връща кодиран символ с номер n (0 ≤ n ≤ 63).