HÁZI FELADAT

Pontosított feladatspecifikáció

Vizi Előd
S53O1S
2016. április 04.

1.Feladat: Az én feladatom készíteni egy objektumot 11 jegyű BCD számok tárolására. Az összes értelmes műveletet ezen az osztályon operátor átdefiniálással (overload) valósítsam meg, viszont nem kell kötelezően minden operátort átdefiniálnom. Majd specifikálnom kell egy egyszerű tesztfeladatot, ami felhasználja ezt az adatszerkezetet.

2.Pontosított feladatspecifikáció: Mivel sokkal jobban szeretjük, legalább is én mindenképp jobban szeretem kezelni a decimális számokat, ezért mindenképp úgy valósítanám meg ezt az adatstruktúrát, hogy lenne két globális függvényem, amelyek meghívásával bármikor át tudom váltani a BCD számomat Decimálisra, valamint fordítva, Decimálisról vissza BCD –re. Továbbá így is valósítanám meg az operátorokat is. Két ilyen BCD összeadása esetén először mindkettőt átkonvertálnám decimálisra, majd úgy összeadom a számokat és végül pedig visszakonvertálnám BCD –re. Így könnyen megvalósíthatóvá válnak azok a műveletek is, amikor a művelet valamelyik oldalán mondjuk nem egy ilyen BCD kódolású szám ál, hanem egy decimálist szeretnénk hozzáadni egy BCD –hez. Viszont a fent leírt konverzió nem valósítható meg integer változók segítségével. Ezért a biztonságos tárolás érdekében „long long int” –et szándékozom használni, hogy a 11 hosszú BCD –ket is tárolni tudjam decimálisként.

A feladat működésének szimulálására egy számológépet szeretnék bemutatni (grafikai felület nélkül) ami kimondottan ilyen adatszerkezetekkel dolgozok. Ez a számológép hatalmas segítséget nyújthat az első féléves Digitális Technika tantárgy hallgatói számára (és nemcsak számukra, hanem majdnem minden informatika érdekeltségű ember számára, akik hasonló problémákba, gondolok én itt BCD számokkal végzett műveletekre, ütköznek munkájuk és tanulmányaik során).

3.Terv: Az én projektem egyetlen osztály felhasználásával megvalósítható, és nem is, hogy csak megvalósítható, hanem ez adja a lehető legegyszerűbb és legátláthatóbb megoldást is. Gondolkodtam azon is, hogy lenne egy osztályom ami a 4 bites szekvenciákat külön kezelné és a fő osztályom ezekből összefogólag kezelne maximum 11-et. De igazából ez a megvalósítás semmivel sem könnyítené meg az én munkámat sem, sem pedig az átláthatóságot, fejleszthetőséget. Így tehát úgy döntöttem, hogy egyetlen osztály segítségével oldom meg a feladatomat.

3.1 Objektum Terv: Ennek az osztálynak az UML diagramja a következő oldalon látható. Ezt a diagramot az „Enterprise Architect” nevű modellező program trial verziójával generáltam, ami letölthető az alábbi linken: http://www.sparxsystems.com/products/ea/trial/request.html.

Fontosnak tartom megjegyezni, hogy ez az osztály még bővülhet egy két operátorral, vagy esetleg egy-két újabb alprogrammal. Próbáltam átgondolni, hogy mikre lesz majd szükségem és az összes lehetséges felhasználandó operátort már most belerakni, de a fejlesztés folyamán adódó esetleges újabb problémák, egyszerűsítések megoldására, lehet szükséges lesz majd új függvényeket is bevezetni.

A legtöbb függvényt „friend” (barát) függvényként hoztam létre, hogy majd elérhetőek legyenek a függvénytörzsben az osztályom privát tagjai is.

3.2 Algoritmusok: A tagfüggvények definiálása és az osztályom létrehozása folyamán pár szükséges és alapvető függvényt már a létrehozás folyamán megírtam, ilyenek például a konstruktor, destruktor, operator=, valamint az indexelő (operator[], inline függvényekként) operátorok. Ezek az algoritmusok alább láthatók. Azonban ezek is még változhatnak a fejlesztés folyamán.

```
#include<iostream>
#include<cmath>
#include "BCD.h"
#define HOSSZ darab*4 //darab 4 szerese, mivel egy tagot 4 bitbe tarol

BCD::BCD(){
    data = NULL;
    darab=0;
}

BCD::BCD(int darab):darab(darab){
    data=NULL;
    if(darab!=0){
        data=new int[HOSSZ];
        for(int i=0;i<HOSSZ;i++){
            data[i]=0;
        }
    }
}

BCD::BCD(const BCD & rhs){
    darab=rhs.darab;
    if(rhs.data==NULL)
        data=NULL;
    else{
        data = new int[HOSSZ];
        for(int i=0;i<HOSSZ;i++){
            data[i]=rhs.data[i];
        }
    }
}

BCD& BCD::operator= (const BCD & rhs){
    if(this!=&rhs){
        delete[] data;
        darab=rhs.darab;
        data=new int[HOSSZ];
        for(int i=0;i<HOSSZ;i++){
            data[i]=rhs.data[i];
        }
    }
    return *this;
}
```

```
int operator[] (size_t idx)const{
    return data[idx];
}
int & operator[] (size_t idx){
    return data[idx];
}
size_t size()const{
    return darab;
}
size_t realsize()const{
    return HOSSZ;
}
```