Informatika szak

Az „Adatszerkezetes” Beadandó Feladat Értékelési Szempontjai

1. A BEADANDÓ FELADAT CÉLJAI

A beadandó feladat célja annak bizonyítása, hogy a hallgató képes önállóan azokat a módszereket és formalizmust alkalmazni, amelyeket a típusok definiálására és megvalósítására el kellett sajátítania, és azokat a számítógépi eszközöket a feladat megoldása során hatékonyan fölhasználnia, amelyekkel megfelelő minőségű munkát képes leadni.

2. BEADANDÓ

1. A komplett anyagot a **moodle rendszerében kell feltölteni egy** tömörített **fájl**lá összecsomagolva. A fájl a következő pontban részletezett struktúrában tartalmazza az anyagot, amelynek **neve**: **feladatsorszám + EHA-kód** (Pl. 01NEUMJANO.zip).
2. A beadandó részei: a **dokumentáció**(ka)t tartalmazó fájl, a **Pascal forráskód**ot tartalmazó fájlok (a beépített saját unit-, include-oké is), valamint a lefordított program (exe-fájl), továbbá a fordításhoz és futtatáshoz esetleg szükséges további fájlok (pl. tesztadat-fájlok). A beadandót alkotó **könyvtárszerkezet**:

```
🖫\          -- a gyökérben a futtatási környezet (EXE + adatfájlok)
  🗀\FORRAS\ -- PAS programállomány + UNIT/INCLUDE forrásállományok
  🗀\DOKU\   -- DOC-állományok
```

3. A **feladat sorszáma**, **szövege** és a **név** a dokumentáció nyitó lapján jól látható helyen szerepeljen! (L. a mintadokumentációt doc-ban, pdf-ben!)
4. A **dokumentáció** kinyomtatott formában is behatató (ekkor tudunk részletesebb értékelést adni róla).

3. SZEMPONTOK

3.1. Módszerek

A ’**Felülről-lefelé programkifejtés**’ elvének, valamint a **programozási tételek** fölhasználásának tükröződnie kell az algoritmuson. (Az alkalmazott tételek nevei szerepeljenek megjegyzésként a megfelelő algoritmikus részletek mellett.) A **feladathoz illeszkedő típusok** definiálása és megvalósítása. A megvalósításnál törekedni kell a típus „újrafelhasználhatóságára”,

azaz a **megvalósító kódot különálló fájl**ban kell elhelyezni. (Egyszerűen szólva: *ahány új típus*t kell definiálni a programhoz, *annyi fájl*ban kell elhelyezni a hozzájuk tartozó kódot.)

3.2. Formalizmusok

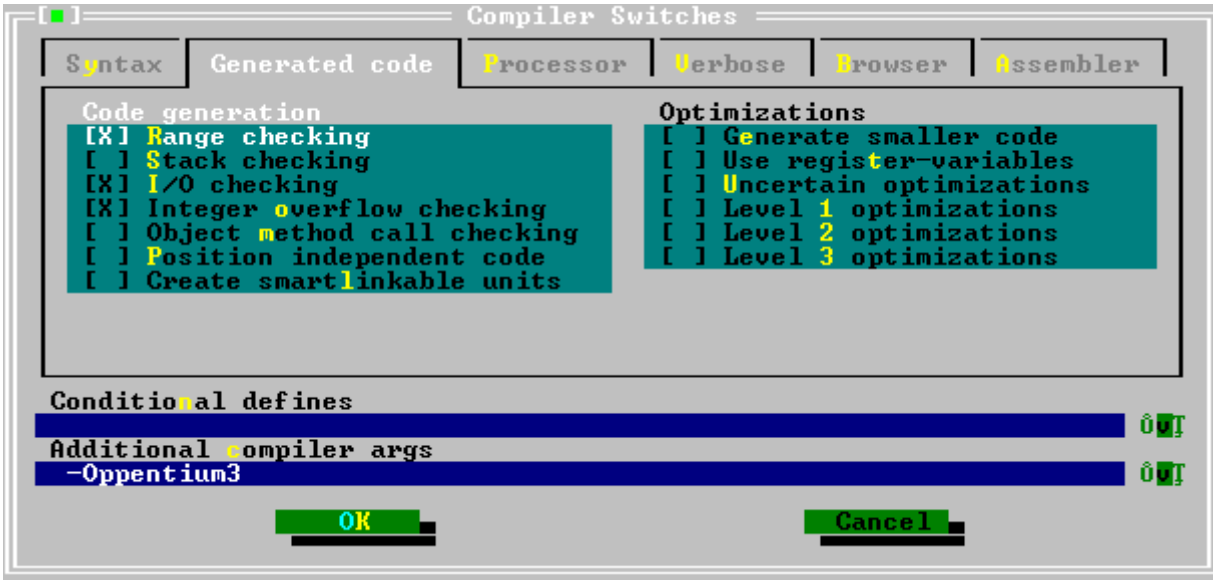
Specifikáció(k) –ideértve a definiált **típusok**ét is– pontos részletezése. Nem muszáj matematikai jelöléseket használni, de teljesnek, egyértelműnek és precíznek kell lennie; az új típusok algebrai leírását nem kell mellékelni, de a **moduljai**t igen. A modulokban el lehet tekinteni mindenegyes operáció elő- és utófeltételeinek részletezésétől. Fontos, hogy legyen egy-két példa az operációk specifikálására is!
Az **algoritmus**t (és a típust definiáló **modulok**at) az órákon is használt pszeudó kód segítségével kell leírni. A kód **FreePascal**-ban készüljön.
A **fejlesztői dokumentáció** rövid, de minden megadott „kérdésre” (pl. kód is!) válaszoljon. A **felhasználói dokumentáció** igényesen (futásokból vett screenshot-okkal illusztrálva) bemutatja a programhasználat mikéntjét.

3.3. Eszközök

FreePascal fejlesztőrendszere (esetleg más hasonló keretrendszer, pl. Geany) a program gépre viteléhez (több fordítási egységből álló program szerkesztése, fordítása és futtatás, elemi hibakeresési szolgáltatások készsége).
Valamilyen igényes számítógépes **szövegszerkesztő** a dokumentáció elkészítéséhez. (Tükröződjön a dokumentáción: szövegszerkesztési rutin; minimális ergonómiai igényesség: címek, fejezetcímek, bekezdések, igazítások alkalmazása; lapsorszámozás; magyarázó, illusztráló ábrák).

4. SÚLYOK

Legnagyobb súlyt az **algoritmus helyessége** mellett az alkalmazott **típusok megvalósításának meggondoltsága** kapja. Ezt követi a program minősége: a barátságosság (képernyőtörlés, beolvasás és eredménymegjelenítés egyértelmű jelzése, gusztusos volta) és a biztonságosság (a program által a bemenő paraméterekkel szemben támasztott elvárásoknak ellenőrzése; hibás fájlinput esetén hiba kijelzése után a program megáll, de a programnak csak egyetlen hiba-megállási pontja lehet).
Szintaktikusan hibás programot érdemben **nem értékelünk**. Szemantikusan hibás (elszálló), vagy nem a megadott adatszerkezet(ek)re épülő, esetleg lényegesen leegyszerűsített feladatot megoldó program töredékértékben számítható csak be. (Utóbbi két esetben teljes pontértékkel csak a dokumentáció számítható, minden más csak bizonyos százalékban.) A kipróbálás előtt a FreePascal OPTION-COMPILE menübeli **Range checking .. Overflow checking** fordítási opciók be kell kapcsolni, s így fordítandó le a program! (L. az alábbi ábrát.)


Compiler Switches
Syntax
Generated code
Processor
Verbose
Browser
Assembler
Code generation
[X] Range checking
[ ] Stack checking
[X] I/O checking
[X] Integer overflow checking
[ ] Object method call checking
[ ] Position independent code
[ ] Create smartlinkable units
Optimizations
[ ] Generate smaller code
[ ] Use register-variables
[ ] Uncertain optimizations
[ ] Level 1 optimizations
[ ] Level 2 optimizations
[ ] Level 3 optimizations
Conditional defines
Additional compiler args
-Oppentium3
OK
Cancel

5. PONTOZÁS

Programkód *futtatással ellenőrizve* **Pr**	Fut, *helyes* eredményeket produkál (használat *kényelmessége*, *egyér-telműsége*,...)	0..30
	Ha nincs legalább 5 –lényegesen eltérő– tesztadatfájl	*összesen: 0*
	Ha *nincs forrás*fájl (**Ps**)	**Ps**=0.5
	Ha *nem fordítható le és EXE sincs*, akkor összpont (utólag pótolnia kell)	*összesen: 0*
Ps*Pr	**Minimum..Maximum**	**0..30**
Dokumentáció	*Ha nincs, akkor nincs specifikáció, nincs algoritmus és nincs ellen-őrizhető kód. Ezért azok pontjai elvesznek.*	*Összesen: 0..20*
D	Fejlesztői teljessége	0..8
	Felhasználói teljessége	0..7
	Igényesség (a teljes dokumentációé; pl. ábrák vannak-e…)	0..5
D	**Minimum..Maximum**	**0..30**
Specifikáció	A feladatnak nem megfelelő specifikáció büntető súlya (**Ss**)	**Ss**=0..1
S	Program specifikáció	0..10
	Bemeneti, kimeneti fájlok szerkezetének részletezése	0..5
	A megadott típusok operációinál szerepel legalább 2 ef/uf	0..5
	Formalizáltság (hatékony, leírást rövidítő fogalmak; matematizáltság)	0..10
Ss*S	**Minimum..Maximum**	**0..25**
Algoritmus	A specifikációtól (a megadott ábrázolástól) eltérés büntető súlya (**As**)...	**As**=0.5..1
	Ha nem típusok bevezetésével oldja meg a problémát, hanem pl. a kód közvetlen beillesztésével (nincsenek MODULok) akkor.....	**As**=0.5
	Algoritmus nincs, a büntető súly (**As**=0.5, és A=0)	**As**=0.5,**A**=0
	Ha nem felülről-lefelé tervez, akkor	A=0
A	Tételek alkalmazása tükröződik az algoritmuson	0..10
	Az eljárásoknak/függvényeknek van elő-, utófeltétele „mutatóban”	0..5
	Típusok körültekintő választása, jól definiálása és megvalósítása	0..10
	Hibákért (pl. nem megengedett szerkezetek...) egyedi levonások	-10..0
Ss*As*A	**Minimum..Maximum**	**0..25**
Kód *(doku. alapján)*	Az algoritmustól eltérés büntető súlya (**Ks**)	**Ks**=0..1
	Ha nem ellenőrizhető a kód, akkor e rész összpontja	**K**=0
K	A típusok megvalósítása (reprezentáció+implementáció)	0..10
	Lapozott fájlos tájékoztató.....	0..5
	Barátságos (beolvasás, visszajelzés, eredménymegjelenítés)	0..5
	Biztonságos (beolvasás, fájlmegnyitás)	0..5
	Fájlos hibafigyelés és –jelzés.....	0..5
	Hibákért (GOTO, EXIT...) egyedi levonás	-5..0
	Több hiba-leállási pont	-2
	Ha egyetlen fájlba „gyömöszölte”…	-10
Ss*As*Ks*K	**Minimum..Maximum**	**0..30**
Pluszokért: Pl	(kódfájlok, ablakos menü, help...)	**0..10**
Beadandó:	**Összpontszám, Minimum..Maximum: (Ps*Pr+D+Ss*S+Ss*As*A+Ss*As*Ks*K+Pl)**	**0..140+10**

6. ÉRTÉKELÉS

Alsóhatár	Felsőhatár	Jegy
120	..	5
100	99	4
80	84	3
60	69	2
..	54	1

Minden megkezdett hét késés egy jegy levonásával jár.

Fontos: a beadandókat szóban is meg kell védeni. Ennek időpontját és helyét egyeztetjük.