Adatbázis és szoftverfejlesztés elmélet

Témakör 3.

Összefoglalás

1. Programtervezési stratégiák:

Egyszerű programjaink készítése során nem okoz különösebb gondot, hogy a programok szerkezetét, a szükséges eljárásokat, adatszerkezeteket megtervezzük: ezek egyszerű esetekben természetesen „adódnak. Bonyolultabb programok esetében a tervezési folyamatot célszerű tudatos módon, szisztematikusan végezni. A gyakorlatban két alapvető gondolkodási módot találunk:

- A felülről lefelé tervezést
- Az alulról felfelé tervezést

Felülről lefelé tervezés (top-down módszer)

A felülről lefelé tervezés alapelve az *analízis*, azaz a célt elemezve a feladatot fokozatosan egyre kisebb logikai egységekre bontjuk. Majd ezeket ismét tovább analizálva még kisebb logikai egységeket kapunk. Az analízis-részekre bontás folyamatot addig folytatjuk, míg már tovább nem bontható, elemi tevékenységekig jutunk.

Előnyei: A tervezés során definiált egységek kapcsolatai, interfészei jól definiáltak, ezért ezek önállóan is kezelhetők, párhuzamosan is fejleszthetők. A kódolás tehát könnyű és áttekinthető, az eredmény jól strukturált, logikus felépítésű lesz.

Hátrányai: A felülről lefelé tervezéssel készült szoftverek tesztelése csak a tervezési és fejlesztési folyamat legvégén kezdődhet el, mert a fejlesztés során felülről lefelé haladva egységeink definíciói olyan alegységekre hivatkozhatnak, melyek egyelőre csak funkcionálisan ismertek, ezek definíciója csak később készül el.

Alulról felfelé tervezés (bottom-up módszer)

Az alulról felfelé tervezés alapelve a *szintézis*, azaz a már meglévő elemekből történő építkezés. Az alulról felfelé építkezés feltétele, hogy az elemi tevékenységeket ismerjük, ezeket kezdetben meghatározzuk. Ezen elemi tevékenységeket kell logikusan rendezni mindaddig, amíg azok megfelelő sorrendjét és hierarchiáját ki nem alakítjuk, és ezzel a programozási feladat megoldását megkapjuk.

Előnyei: Az alulról felfelé tervezés nagyon jól támogatja a már létező szoftverkomponensek újrafelhasználását. A tervezés és implementálás már működő komponensekből történik, így a létrehozott összetett komponensek azonnal ki is próbálhatók, tesztelhetők. Mire a tervezési folyamat végére érünk, egy gyakorlatilag futtatható rendszer áll rendelkezésünkre.

Hátrányai: Rosszul definiált elemi egységek egymásra építése kevésbé logikus programot eredményezhet. Egyes esetekben a már létező komponensek utólagos egymáshoz illesztése komoly feladat lehet.

2. Kódszöveg szerkesztése:

Karakter - A kódszöveg legelemibb alkotórésze. Megjelenései az adott kódtáblától is függ.

- betűk
- számjegyek
- egyéb karakterek pl.: \ / * + @ $

A program lexikális egységei

- azonosítók
- fenntartott szavak
- címkék
- elválasztó jelek
- megjegyzések
- változók
- konstansok

Azonosítók
A programban saját objektumainknak valamilyen nevet adunk azonosítás céljából. Az azonosítók jellemzői: általában betűvel kezdődik, betűvel vagy számmal folytatódhat és saját objektumaink megnevezésére használjuk.

Fenntartott szavak

- A nyelvben már valamilyen céllal felhasznált nevek
- azonosító jellegűek és felépítésűek
- a nyelv rendel hozzájuk jelentést
- csak az adott célra használhatók

Címkék

- utasítások azonosítására használható
- valahol számmal is kezdődhet
- lehet azonosító jellege

Elválasztó jelek
A programozás történetének kezdeti szakaszában az "egy sor-egy utasítás” szerkezet volt jellemző. Napjainkban a kód szerkesztése szabad formátumú az elválasztó jelek használatának köszönhetően. Utasítás szeparátorok (elhatároló jelek) pl.: ; , : ( ){}

Megjegyzés
A megjegyzések nem a fordítónak vagy az interpreternek szólnak, hanem a programozóknak nyújtanak segítséget. Saját programjainkat gyakran látjuk el megjegyzésekkel, hogy később gyorsan kiderüljön mit is csinál az adott kód, vagy a tesztelés során is hasznos lehet használatuk.
Példák:
PL/1, C, C++, C#, Java:
/* több soros megjegyzés */
//megjegyzés
Pascal:
{ megjegyzés }
Shell, Perl:
megjegyzés

Néhány programozási nyelvben csak egy soros megjegyzések alkalmazhatók, valamelyik többsorosat is megenged.

Változó
A változók olyan memóriaterületek, amelyeken valamilyen értéket tárolunk, de azok értékét bármikor megváltoztathatjuk.
Egy változó felépítése: *név – típus – érték - memóriacím*

Konstansok
A konstans szintén egy a memóriában lefoglalt terület egy érték számára. A konstansok esetén a programozó azonban azt vállalja, hogy annak értékét a továbbiakban nem változtatja meg.
Konstansok osztályozása:

- nevesített konstans
- literális konstans
-

Nevesített konstansok felépítése: *név – típus - érték (címkomponens hiányában nem változtatható meg)*
Literális konstansok felépítése: *típus – érték (A literálnak tehát nincs neve. Az értéket egyszerűen leírjuk.)*
A számokat általában önmagában, a karaktersorozatokat általában idézőjelek, vagy aposztrófok között. A literális lkonstans önmagát deklarálja. A programba fix értékkel kerülnek. Például: 5, 2.5, 4., "alma"
Az első példa egy egész típusú literális konstanst határoz meg. A második és a harmadik egy valós típusú konstanst. A negyedik egy karaktersorozat típusú konstanst.

3. Adattípusok

Értékhalmaz szempontjából az adat lehet:

- *Elemi típus:* Szerkezetileg nem bontható tovább, például: egész, valós, logikai, karakter, mutató (adat címe), felsorolás, intervallum.
- *Összetett típus:* Szerkezettel rendelkezik, például: rekord (különbözőtípusú, de logikailag összetartozó típusok együttes kezelése), halmaz, szöveg (karakterek sorozata), sorozat, tömb (mátrix).

4. Elemi adattípusok

Az elemi adattípusok legfontosabb jellemzői:

Egész
: Értékhalmaza: pozitív és negatív egész számok és 0, az intervallum az ábrázolástól függő (például 2 byte esetén: -32768 .. 32767)

Valós
: Értékhalmaza: valós számok, intervalluma szintén az ábrázolástól függ, többféle értékhalmazzal szokás realizálni

Logikai
: Értékhalmaza: igaz, hamis (true, false)

Karakter
: Értékhalmaza: 0..255 kódú jelek halmaza

Felsorolás

Értékhalmaza: konstansok felsorolásával adjuk meg

Intervallum

Értékhalmaz: az alaptípus (más elemi típus) összefüggő részhalmaza

Mutató

Értékhalmaz: értékek memóriabeli címei, nem negatív egész

5. Összetett adattípusok

Az értékhalmaz mellett a szerkezete (struktúrája) is lényeges.

Rekord

Értékhalmaz: Akkor használjuk, ha különböző alaptípusú, de logikailag össze tartozó adatokat szeretnénk kezelni.

Példa: Típus tanuló=Rekord
(név: szöveg;
kor: egész)

Halmaz

Értékhalmaz: Alaptípusba tartozó elemek sora, amely rendelkezik a halmaz tulajdonsággal (minden elem különböző, nincs sorrendiség)

Szöveg (karakterlánc)

Értékhalmaz: karakterek sorozata, lehet fix hosszúságú, vagy változó.

Sorozat

Értékhalmaz: alaptípusba tartozó értékek sora és a lineáris rendezettség jellemzi: pontosan egy megelőző és következő érték létezik. Tulajdonképpen egydimenziós tömb.

Tömb

Értékhalmaz: alaptípus iteráltja és a lineáris rendezettség jellemzi: pontosan egy megelőző és következő létezik. Több dimenziója is lehetséges.

Kérdések:

1. Mi az alapelve az alulról felfelé történő tervezési folyamatnak!
2. Ismertesse felülről lefelé történő tervezés alapelvét!
3. Sorolja fel a programkód lexikális egységeit!
4. Mi jellemzi az azonosítókat?
5. Lehetnek-e változónevek a fenntartott szavak?
6. Mik a szeparátorok?
7. A megjegyzések milyen szereppel bírnak a programban?
8. Ismertesse a változók felépítését!
9. Mutassa be a konstansok fajtáit és azok felépítését!
10. Mi az alapvető jellemzője az elemi adattípusoknak? Soroljon fel 3 ilyen típust!
11. Mi az alapvető jellemzője az összetett adattípusoknak? Soroljon fel 3 ilyen típust!