Programozási nyelvek I.
2. gyakorlat

Balogh Ádám
bas@elte.hu
Eötvös Loránd Tudományegyetem
Informatikai Kar

Segítség az 1. feladathoz (1)

- Pozitív egészek típusa: `Positive`
- Karakterlánc változók deklarálása:
  `Név : String ( Eleje .. Vége );`
- Kiírás soremelés nélkül: `Text_IO.Put`
- Beolvasás billentyűzetről:
  `Text_IO.Get_Line ( Karakterlánc, Hossz )`
- Karakterlánc részlánca:
  `Karakterlánc ( Eleje .. Vége )`

Segítség az 1. feladathoz (2)

- Karakterlánc konvertálása pozitív egésszé:
  `Positive'Value ( Karakterlánc )`
- Elágazás diszkrét kifejezés értéke alapján:

```
case Kifejezés is
   when Értékhalmaz =>
      Legalább egy utasítás
   when Értékhalmaz =>
      Legalább egy utasítás
   ...
end case;
```

Segítség az 1. feladathoz (3)

- Értékhalmazok: *Érték*, *Érték* `..` *Érték*, *Értékhalmaz* `|` *Értékhalmaz* és `others`
- Az egyes ágak értékhalmazainak diszjunktaknak kell lenni
- Az összes ág értékhalmazai uniójának le kell fednie a kifejezés típusának típusértékhalmazazát
- Az `others` csak a végén szerepelhet, és lefedi a maradék ágakat

1. feladat

Írj Ada programot, ami egy adott sorszámú napról megmondja, hogy az a hét első, utolsó vagy középső munkanapja, egyéb hétköznap vagy hétvége. Ha a nap sorszáma nagyobb, mint 7, a program adjon hibaüzenetet. Használd a `case` utasítást! A program neve legyen `Naptar`!

Segítség az 2. feladathoz (1)

- Függvény definiálása:

```
function Név ( Paraméterek ) return Típus is
   Változódeklarációk
begin
   Utasítások
end Név;
```

- Paraméterek alakja:

```
Név, Név, … : Típus; Név, Név, … : Típus; …
```

- Külön fordítási egységben: *név*`.adb`
- Paraméterek értéke nem változtatható
- Érték visszaadása: `return` *Kifejezés*

Segítség a 2. feladathoz (2)

- Függvény használata a főprogramban: mint egy csomagot: `with` *`Név`*, de nem kell (és nem is szabad) utána `use` *`Név`*!
- Függvény meghívása: tetszőleges (függvény visszatérési típusának megfelelő kifejezés helyén): *`Név`* `(` *`Paraméterek`* `)`
- Paraméterek itt: *`Kifejezés`*, *`Kifejezés`*, …, amiknek típusa meg kell, hogy feleljen a függvény definíciójában meghatározottnak

2. feladat

Módosítsuk a (javított) lineáris keresést úgy, hogy a találatot egy adott pozíción a `Talalt` függvény jelezze egy logikai típusú érték visszaadásával, melynek három paramétere a szöveg, amelyben keresünk, a karakter, amit keresünk és a pozíció, amelyik után az egyezést szeretnénk vizsgálni. A program neve legyen `Lin_Ker3`!

Segítség a 3. feladathoz

- Függvények be is ágyazhatók egy másik fordítási egységbe (pl. főprogramba):

```
procedure Fő is
   Függvénydefiníciók
   Változódeklarációk
begin
   Utasítások
end Fő;
```

- Ilyenkor nem szerepel a `with` utasításnál!

3. feladat

Ágyazzuk be a `Talalt` függvényt a főprogramba! Az új program neve legyen `Lin_Ker4`!

Segítség a 4. feladathoz

- Egy változó deklarációja az adott blokkban, és annak a deklarációt követő alblokkjaiban látható
- Adott blokkra nézve globális változó: a blokkot tartalmazó blokkok valamelyikében van deklarálva
- Függvénydefiníciók és változódeklarációk keverhetők tetszőleges sorrendben egy deklarációs részen belül

4. feladat

A begyázott `Talalt` függvény ne kapjon paramétereket, hanem a főprogram változóit, mint globális változókat használja! Az új program neve legyen `Lin_Ker5`!

Segítség az 5. feladathoz

- Rekurzív függvény: önmagát hívja
- Vigyázni kell, nehogy „végtelenszer” hívja önmagát

5. feladat

Alakítsd át a 10 faktoriálisát számoló programot úgy, hogy a faktoriálist a `Fakt` rekurzív függvény számolja, melynek paramétere az a szám, amelynek faktoriálisára kíváncsiak vagyunk. Ez a függvény lehet külön fordítási egység vagy akár a főprogramba ágyazott függvény is. Az új program neve legyen `Faktor2`!

Segítség a 6. feladathoz (1)

- Eljárás definiálása:

```
function Név ( Paraméterek ) is
   Változódeklarációk
begin
   Utasítások
end Név;
```

- Paraméterek alakja:

```
Név, … : Mód Típus; Név, … : Mód Típus; …
```

- Egy paraméter módja lehet bemenő (`in`), kimenő (`out`) vagy ki- és bemenő (`in out`)

Segítség a 6. feladathoz (2)

- Eljárás ugyanúgy lehet külön fordítási egység, mint egy függvény, és ugyanúgy beágyazható más fordítási egységbe is
- Ugyanúgy with-tel use nélkül használjuk, mint egy függvényt
- Meghívása: utasítás helyén, ugyanúgy, mint egy függvényt
- Az `out` és `in out` paraméterek helyére csak változó írható!

6. feladat

Írj egy `Kozos` eljárást, amely kiszámítja két pozitív egész legkisebb közös osztóját és legnagyobb közös többszörösét. Az eljárás bemenő paramétere legyen a két szám, kimenő paramétere pedig a két eredmény. Az eljárás segítségével írj programot, amely kiszámítja 12 és 18 legnagyobb közös osztóját és legkisebb közös többszörösét. A program neve legyen `LNKO_LKKT`!

7. feladat

Írj egy `Novel` nevű eljárást, amely megnöveli a paraméterben kapott változó értékét eggyel! Írj programot az eljárás segítségével, amely megnöveli egy kezdetben 6-ot tartalmazó változó értékét, és kiírja a képernyőre! A program neve legyen `Noveles`!

Segítség a 8. feladathoz

- Paramétereknek lehet alapértelmezett értéke: *Név, … : [in] Típus := Érték*
- Ezek elhagyhatók (de nem középről)
- Hívás névvel jelölt formában: paramétereket *Név => Kifejezés* alakban adjuk meg, sorrendtől függetlenül
- Vegyes forma: előbb pozicionális, majd névvel jelölt

2004. szeptember 23. Programozási nyelvek I. - 2. gyakorlat 34

8. feladat

Egészítsd ki a `Novel` eljárást `Novel2` néven egy bemenő paraméterrel, amely azt mondja meg, hogy a számot mennyivel kell megnövelni. A paraméter legyen elhagyható, ebben az esetben az eljárás ugyanazt a műveletet végzi, mint a `Novel`. A `Noveles2` programban próbálj ki minél több paraméterátadási formát, amikor a `Novel2`-t hívod!

2004. szeptember 23. Programozási nyelvek I. - 2. gyakorlat 35

Házi feladatok (1)

- Írj egy `Max2` és egy `Max3` nevű függvényt, amely két illetve három szám maximumát számolja ki. A két főprogram, amely ezeket használja legyen `Maximum2` és `Maximum3`, és írja ki a képernyőre 4 és 6 illetve 4, 6 és 11 közül a legnagyobbat!
- Ágyazd be mindkét függvényt azonos `Max` néven egy `Maximum` nevű főprogramba, amely egymás után kiírja 4 és 6, illetve 4, 6 és 11 maximumát!

2004. szeptember 23. Programozási nyelvek I. - 2. gyakorlat 38

Házi feladatok (2)

- Írj programot, amely összeszorozza egymással az első két parancssori argumentumában kapott számot! Ha az argumentumok száma kevesebb, mint kettő, írjon ki hibaüzenetet a képernyőre! A program neve legyen `Szoroz`! (Segítség: `Ada.Command_Line`, könyv A.6.)