Tartalomjegyzék

- 1 A programozás alapjai
  - 1.1 A számítógép felépítése
  - 1.2 Eljárás, algoritmus, programozás
  - 1.3 Változók, adatok, adattípusok
    - 1.3.1 Változók definiálása és deklarálása
      - 1.3.1.1 Röviden a láthatóságról
- 2 Ellen?rz? kérdések
  - 2.1 Források

A programozás alapjai

A számítógép felépítése

A számítógép f? részegységei: Kép

- processzor (beolvassa a memóriából az utasításokat és az adatokat, az utasítások alapján m?veleteket végez, az eredményt visszaírja a memóriába; valamint vezérli a perifériákat - adatokat olvas bel?lük, ill. ír ki)
- memória (általános tároló, mely utasításokat és adatokat tartalmaz)
- perifériák (pl. háttértároló (ahol az elmentett fájlok találhatók); beviteli eszközök: billenty?zet, egér; megjelenít? eszközök: monitor)

Eljárás, algoritmus, programozás

- **Programozás**: el?írni a számítógépnek, hogy mit csináljon. (Ebben a félévben csak *imperatív* programozás: lépésr?l lépésre el?írjuk, hogy mit kell tenni, kb. mint egy recept leírása.)
- **Eljárás (módszer)**: Gépiesen (gondolkodás nélkül) végrehajtható lépések sorozata, amely elvezet a megoldáshoz.
- **Algoritmus**: Gépiesen (gondolkodás nélkül) végrehajtható lépések **véges** sorozata, amely elvezet a megoldáshoz.

Miel?tt az algoritmust kódolni kezdjük, meg kell gy?z?dni róla, hogy mindhárom feltétel teljesül:

- *helyes* : Valóban azt a feladatot oldja meg, amire szánjuk.
- *teljes* : Minden lehetséges esetben megoldja.
- *véges* : Véges sok lépésben befejez?dik.

Példa *módszer*-re: Newton módszer

Példa *algoritmus*-ra: bináris keresés
Egy programozási feladat általános megoldási folyamata: Kép

Változók, adatok, adattípusok

- **Változó**: Az olyan adatot, amelyet meg lehet változtatni, változónak nevezzük. (pl: *int x;*)
- **Konstans**: Az olyan adatot, amelynek értéke nem változtatható meg, konstansnak nevezzük. (pl: *12*)

A C er?sen típusos nyelv, ráadásul statikusan típusos, azaz a változókról meg kell mondani, hogy milyenfajta adatok tárolására alkalmasak. Például *int* jelöli azt, hogy egész számokat tárolhat, *float* pedig azt hogy lebeg?pontos tört számot.

(A Python dinamikusan típusos, ott az értéknek van típusa, nem a változónak. Ezért nem kellett definiálni a változókat, azokban bármilyen adat lehetett. Ez egyszer?síti a programírást, viszont növeli a hibalehet?séget.)

Változók definiálása és deklarálása

- **Deklaráció**: A deklarációkor egy adott dologról jelezzük a fordítónak, hogy létezik, de ott még nem fejtjük ki azt.
  - a változó típusát, méretét (alapvet? jellemz?it) határozza meg
  - utána a deklarált azonosító használható a kódban
  - Példa: két változó (egy egész és egy lebeg?pontos típusú) deklarálása:

```
int alma;
 float barack;
```

- **Definíció**: Definíciókor egy adott dologról pontosan megmondjuk, hogy micsoda. Ez változók esetén azt jelenti, hogy már értéket is kap.
  - meghatározza valamely objektum típusát, méretét
  - hatására helyfoglalás is történik a memóriában
  - Példa: két változó (egy egész és egy lebeg?pontos típusú) definiálása:

```
int alma = 3;
 float barack = 2.6;
```

Megjegyzések:

- A definíció egyben deklaráció is.
- Kés?bb lesz szó egyéb dolgok (pl. függvények) deklarációjáról és definíciójáról is.

Röviden a láthatóságról

Egy változó *láthatóságán* azt értjük, hogy a programkód mely részeib?l érhet? el, hol használható. Az egyszer? szabály az, hogy a változó azon a blokkon belül érhet? el, ahol deklaráltuk. Itt blokknak számít egyrészt maga a teljes c-forrásfájl (ami egyel?re a teljes programunkat jelenti) (az itt deklaráltak globális változók lesznek), másrészt minden kapcsos zárójelek közötti rész egy-egy újabb blokk (ezen belül lokális változóink lesznek).

Például a következ? kódrészlet le sem fordul:

```
int x = 2; /* ez egy globális változó, mindenhol használható a programban */
```

```
int main(void) {
    if (x > 5) {
        int y = 3; /* lokális változó, csak a következ? } -ig él */
    } else {
        y = 5;  /* HIBA: ebben a blokkban nem látszik a fentebbi if-es blokkban definiált y !*/
    }
    return 0;
}
```

Többféle módon is kijavíthatjuk a fordítási hibát.

Az els? esetben egy *y* nev? **globális** változót veszünk fel az *x* mellé:

```
int x = 2;
int y; /* globális változó */
int main(void) {
    if (x > 5) {
        y = 3;
    } else {
        y = 5;
    }
    return 0;
}
```

(Itt arra kell még figyelni, hogy ha véletlenül ottmarad az "http://wiki.math.bme.huint"http://wiki.math.bme.hu szó az 5. vagy 7. sorban, akkor ott egy új, lokális *y* változó fog létrejönni, és abban a blokkban az új lesz az érvényes, nem a küls? *y*. Az azonos nev? változók közül mindig a "http://wiki.math.bme.hulokálisabb"http://wiki.math.bme.hu az érvényes.)

Igazából a problémát az is megoldja, ha az *y* nem globális lesz, csak egy blokkal kintebb visszük (a *main()* függvénybe), hogy az *if* és az *else* ágban is elérhet? legyen:

```
int x = 2;
int main(void) {
    int y; /* függvényen belüli, lokális változó, de kintebbi blokkban */
    if (x > 5) {
        y = 3;
    } else {
        y = 5;
    }
    return 0;
}
```

Egy másik megoldás pedig két, egymástól független lokális változó felvétele, amiknek csak "http://wiki.math.bme.huvéletlenül"http://wiki.math.bme.hu ugyanaz a nevük.

```
int main(void) {
    int x = 2;
    if (x > 5) {
        int y = 3;
    } else {
        int y = 5;
    }
    return 0;
}
```

Egyel?re a globális változós megoldás t?nhet szimpatikusabbnak, mert az a legegyszer?bb, mindenhol használható a változó és kész. De kés?bb, ha majd függvényeket is tudunk írni, jobb lesz a változókat a

függvényeken belül, lokálisan deklarálni. Egy nagyobb programnál a globális változók áttekinthetetlenné és rugalmatlanná teszik a program felépítését. De err?l majd kés?bb részletesebben.

Ellen?rz? kérdések

(Ilyesmi, vagy akár konkrétan ezek várhatóak a gyakorlat eleji mini Zh-n !)

- Rajzold le a számítógép egyszer?sített felépítését (3 doboz)!
- Mi az a három dolog amit ellen?riznünk kell, miel?tt egy algoritmust kódolni kezdünk?
- Mi az "http://wiki.math.bme.hueljárás"http://wiki.math.bme.hu definíciója? Miben különbözik ett?l az algoritmus definíciója?
- Definiálj C nyelven egy *x* nev? és *int* típusú változót, valamit deklarálj egy *y* nev? *float* típusú változót!

Források

- Vitéz András, Dr. Zsóka Zoltán: A Programozás alapjai 1. c. tárgy jegyzete, BME VIK, http://www.hit.bme.hu/~vitez/Progalap1/2011osz/Ea/ea02.pdf
- Pohl László: A programozás alapjai http://www.scribd.com/doc/55046873/h-jegyzet
- Calmarius blogja: A C++ alapjai 11. http://calmarius.net/2011/12/25/basics-of-c-11-source-files-declarations-and-definitions/?lang=hu