A programozás alapjai

Változók

A számítógép az adatokat változókban tárolja

- A változókat alfanumerikus karakterlánc jelöli.
- A változóhoz tartozó adat tipikusan a számítógép memóriájában tárolódik, szekvenciálisan, az adattáblában.
- Némelyik programozási nyelv a változókat egy külön karakterrel jelöli (pl. $, @, %), de pl. a C, C++ nem alkalmaz ilyen konvenciót
- A változó a típusától függő mennyiségű helyet foglal el a memóriában: egész (integer) tipikusan 4 byte, a dupla pontos (double) 8 byte, a karakterlánc a karakterlánc hossza (plusz 1) byteot, a bonyolultabb objektumok az objektum nagyságának megfelelő helyet.
- Minden változóról a programnak tudnia kell a típusát (mennyi adatot kell az adattáblából kiszedni). Néhány nyelv automatikus változófelismerést alkalmaz egy nagyos véges mennyiségű változótípussal.
- Speciális változó: mutató. Ez egy memóriacímre mutat, ahol a hozzá tartozó adat kezdődik. A mutató általában nem mondja meg az adat típusát, ezt meg kell mondani a programnak.

- Lehetnek „konstans” változók, melyek értékét a program nem változtathatja meg.
- A programnak lehetősége van új adattömböket létrehoznia, ezek menedzsmentje az operációs rendszer feladata („szivárog a memória”).
- A függvényhívások során a függvény argumentuma(i) felkerülnek az ú.n. stack-re. A stack egy „verem”, a memória egy jól meghatározott helyén, melybe egymás után betehetünk adatokat, és kivehetünk onnan, fordított sorrendben (az először betett adat jön ki utoljára). 
- A meghívott függvény kiszedi a megfelelő számú argumentumot (ezért a hívó eljárásnak, ill. a függvénynek konzisztensnek kell lennie), majd a végeredményt visszahelyezi a stackre. Pl.

```
int add(int a, int b) {
  return(a+b);
}
```

- Az ilyen típusú függvényhívás tulajdonsága, hogy a változó függvénybeli változtatása nem hat vissza a változó értékére a hívott eljárásban. Pl.:

```
c = dupla(a);
int dupla(int a) {
  a =* 2;
  return(a);
}
```

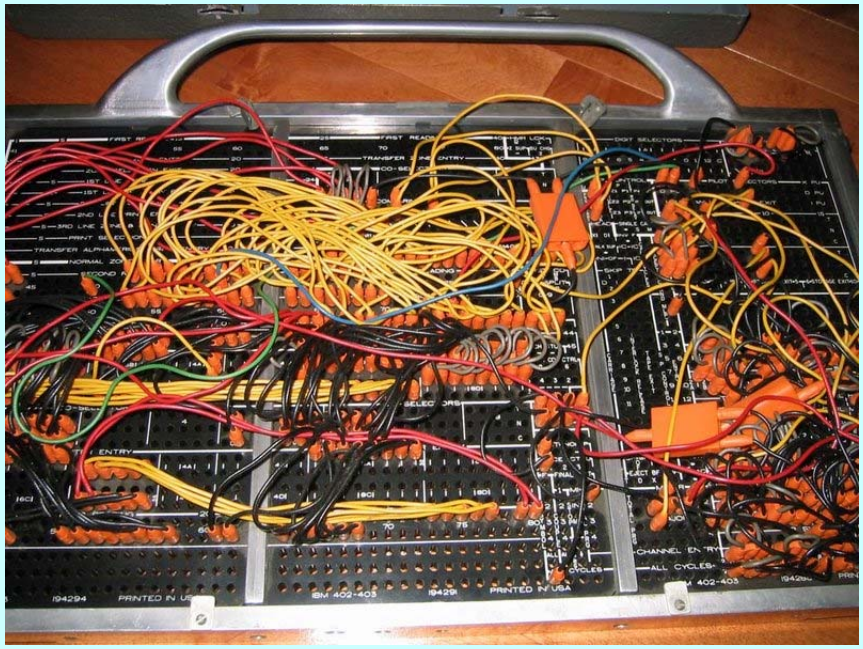
- A nagy programokat csapatok fejlesztik, modulokban. Ezért fontos, hogy a változók ne „akadjanak össze”:
  - Lokális változók: csak az adott függvényben értelmezettek:

```
a=5;                  // lokális változó a hívó eljárásban
c = dupla(a);
…
int dupla(int x) {
  int a=2;       // lokális változó a hívott eljárásban
  x =* a;
  return(x);
}
```

  - Globális változók: a teljes programban értelmezettek, és lehetővé teszik az adatmegosztást a különböző függvények között.

A programozás története

- Az első ismert programozható gépet 1801-ben alkották meg és a szövőgépek mintáját vezérelte. Ezt fejlesztette tovább Herman Hollerith az IBM lyukkártya megalkotásával (US Statisztikai Hivatal)
- A kép egy áthuzalozással programozható gépet mutat
- A Neumann arhitektúra kialakulásával lehetővé vált a program memóriában való tárolása. Az első programok assembly nyelven (gépi kód) íródtak.



IBM 402 programozható könyvelőgép

- Később kialakulnak a magasszintű nyelvek, melyek általában fordítóprogramot igényelnek.
- A rövidebb feladatok megoldására kitalálják az interpreter nyelveket.

	interpreter	fordítandó
szintaxis	laza	Kötött
sebesség	lassú	Gyors

Vezérlés

Alapestben a program szekvenciálisan, az egymás után megadott utasítások végrehajtásával hajtódik végre (egy szál esetében). Ezt a sorrendet a vezérlési utasítások tudják megváltoztatni. A különböző programozási nyelvek ezt különféleképpen valósítják meg, de általánosan a következő típusokat különböztetjük meg:

- A program folytatása egy másik utasításon (ugrás)
- Egy utasítássorozat végrehajtása adott feltétel teljesülése esetén (választás)
- Egy utasítássorozat ismételt végrehajtása (ciklus – az utasítássorozat egy korábbi elemére való ugrás)
- Egy távoli utasítássorozat végrehajtása, mely után a vezérlés visszatér a hívó pontra (szubrutin / függvény)
- A program leállítása (halt)

- Az interrupt is a szubrutinhoz hasonló módon változtatja meg a vezérlést, de ezt nem a programból hívható vezérlési utasítással éri el!
- Egyéb lehetőség még az önmagát átíró kód (csak profiknak).
- A gépi kódban a legtöbbször csak feltételes, illetve feltétel nélküli ugrás áll rendelkezésre, a fordítónak kell a fenti típusokat erre visszavezetni.

Ugrás

- A GOTO utasítás a megadott címkéjű sorra ugrik:

```
goto abrakadabra;
…
abrakadabra: i=0;
```

- A GOTO utasítás és a címke helyzete tetszőleges (előre, illetve hátraugrás).
- A GOTO általában valamilyen feltételhez van kötve.
- A címke pontos alakja függ a programozási nyelvtől (BASIC vagy FORTRAN esetében szám, C, stb. esetében azonosító).

Választás

- Az IF utasítás kiértékeli az utána következő logikai kifejezést, és IGAZ érték esetén végrehajta a logikai kifejezést követő utasítás(sorozato)t:

```
if (valasz eq ”igen”) then
  print ”Ez jó választás”;
elsif (valasz eq „nem”) then
  print ”Sajnálom”;
elsif (valasz eq „talan”) then
  print ”Látom, óvatos vagy”;
else
  print ”Nem tudom értelmezni a választ”;
end if;
```

- A C nyelv nem ismeri az elsif (else if) szerkezet, viszont van benne switch:

```
switch (valasz){
  case ’i’:printf(”Ez jó választás\n”); break;
  case ’n’:  printf(”Sajnálom\n”); break;
  default: printf(”Nem tudom értelmezni a választ”;
}
```

- A logikai kifejezés helyett szerepel aritmetikai kifejezés is (0: hamis, nem nulla: igaz)

Ciklus

- A FOR utasítás a ciklusszámlálóra alapozott ciklust valósít meg:

```
for(i=0;i<1000;i++){
    …
}
```

- A WHILE utasítás feltételre alapozott ciklust valósít meg:

```
i=0; while(i<1000){
    …; i++;
}
```

- A CONTINUE utasítás a cikluson belül a következő iterációra ugrik, a BREAK pedig kilép a ciklusból:

```
for(i=0;i<1000;i++){
   …
   If(i>100) continue;
   …
   If(i==666) break;
   …
}
```

➢ Ha a ciklusszámlálóra alapozott ciklusnál a ciklus belsejében átírjuk a ciklusváltozót, vagy a feltételre alapozott ciklusnál a feltétel sohasem lesz hamis, végtelen ciklusra jutunk.

Szubrutin

- Subroutine, függvény, eljárás, metódus: gyakran használt programrészletek külön blokkba rendezése.
- A szubrutinok paraméterrel láthatók el (bemenő adatok), és visszatérési értéket adhatnak.
- Néhány programozási nyelv megengedi a rekurziót (stack-en átadott változókkal!)
- A szubrutin átláthatóbbá testi a programot, könnyebb az ellenőrzés, fejlesztés, de kissé lassítja a program futását. Néhány fordító ezért hajlamos a szubrutin „befordítani” a fő programba (növelve ezzel a méretét, de csökkentve a futási időt).
- Speciális szubrutinok a kivételek (exception): ezek olyankor futnak le, amikor valami előre definiált esemény következik be (nullával való osztás, érvénytelen művelet – pl. negatív szám gyöke, túlcsordulás)

Kilépés

- A programok az utolsó sorhoz érve leállnak.
- Egy program hamarabb is leállhat:
  - Tervezett módon (exit valamilyen feltétellel pl.)
  - Váratlan módon (le nem kezelt kivétel történik).