Algoritmizálás + kódolás
C++ nyelven és Pascalban

Motiváció

A Programozási alapismeretek tárgyban az algoritmizáláshoz **struktogram**ot, a kódoláshoz **C++** nyelvet használunk, az Algoritmusok és adatszerkezetek c. tárgyban (a közoktatásban elterjedt szokásoknak megfelelően) magyar kulcs-szavú **pszeudokód**ot, illetve (Free)**Pascal** nyelvet.

E dokumentum célja megkönnyíteni a Programozási alapismeretek tárgyban megszerzett ismeretek elehető legkönnyebb konvertálását az újabb algoritmikus és kódolási miliőbe. Bár azt hiszem, hogy más programozási szokásokhoz szokottak számára sem lesz fölösleges legalább egyszer átfutni az anyagon.

Megjegyzem: alábbiakban teljességre nem törekszem, megelégszek azzal, hogy az egyik nyelv ismerete esetén a másik nyelvben könnyen lehessen a kezdő lépéseket megtenni.

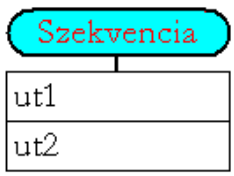
Algoritmizálás: struktogram / pszeudokód

Értékadás, logikai operátorok, inkrementálás és hasonló operátorok

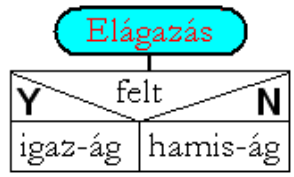
Itt nem a különbözőség, hanem a nyomaték kedvéért sorolom föl őket. Visszatérő jelölés: *vált* – változó; *kons* –konstans; *kif* – kifejezés (azaz formula, képlet)…

Mi fene	Műveleti jel
értékadás	**:=**
értékazonosság	**=, ≠**
rendezés	**<, ≤, ≥, >**
inkrementálás (növelés x-szel)	**:+**x
dekrementálás (csökkentés x-szel)	**:−**x
logikai	**És, Vagy, Nem**
I/O	**Be:** *vált* **Ki:** *kif*

Szekvencia

Struktogram	Pszeudokód
	utasítás$_1$; utasítás$_2$ *vagy* utasítás$_1$ utasítás$_2$

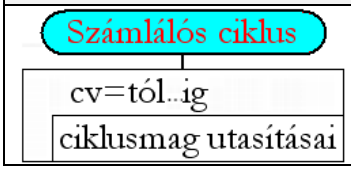
Elágazás

Struktogram	Pszeudokód
	**Ha** *felt* **akkor** *akkor-ág* **különben** *hamis-ág* **Elágazás vége**

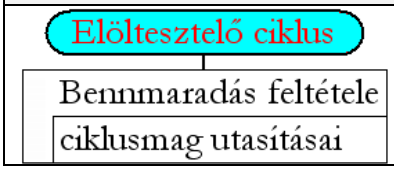
Többirányú elágazás

Struktogram	Pszeudokód
	**Elágazás** *felt*$_1$ **esetén** *utasítások*$_1$ *felt*$_2$ **esetén** *utasítások*$_2$ … **egyéb esetben** *ut* **Elágazás vége**

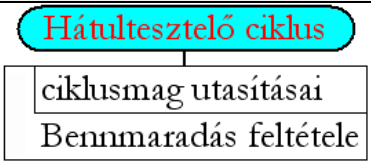
Számlálós ciklus

Struktogram	Pszeudokód
	**Ciklus** *cv*=*k*-**től** *v*-**ig** *cm* **Ciklus vége**

Elölfeltételes ciklus

Struktogram	Pszeudokód
	**Ciklus amíg** *felt* *cm* **Ciklus vége**

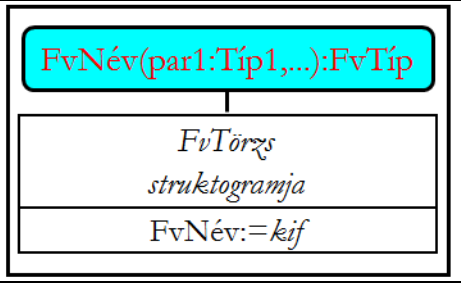
Hátulfeltételes ciklus

Struktogram	Pszeudokód
	**Ciklus** *cm* **amíg** *felt* **Ciklus vége**

Finomításdefiníciók

Fontos egységesíteni a szóhasználatot:

- **típusdefiníció**ról beszélünk akkor, amikor egy addig nem létező **típusnév**hez rendelünk egy memóriába képzési **szabály**t;
- **típusdeklaráció**ról meg akkor, amikor egy **adat**hoz (akár változóhoz, akár konstanshoz) **típus**t rendelünk, ennek során az adott adathoz a típusnak megfelelő mennyiségű **memória** rendelődik (vagy a fordítás során rögvest, vagy majd a végrehajtás során).

Struktogram	Pszeudokód
	**Eljárás** *EljNév(form.par.):* *típusdeklarációk (lokális)* *törzs* **Eljárás vége.**
	**Függvény** *FvNév(form.par.):*FvTíp *típusdeklarációk (lokális)* *törzs* *FvNév:=kif* **Függvény vége.**

Kódolás: C++ / (Free)Pascal

Programszerkezet

Fontos egységesíteni a szóhasználatot:

- **Finomítás** alatt saját eljárást, függvényt vagy operátort értünk;
- **Finomításdefiníció**ról beszélünk, amikor egy finomítás szemantikáját definiáljuk, azaz amikor megadjuk annak működését.

C++	Pascal
`#include <iostream>` `#include <stdlib.h>` `#include includeNév3` … `using namespace std;` *típusdefiníciók* *típusdeklarációk (programparaméterek)* *//akár const-tal bevezetett konstans*	**Program** *PrNév;* **Uses** Crt, *unitNév2,… ;* **Type** *típusdefiníciók* **Var / Const** *típusdeklarációk (programparaméterek)*

finomítás-fejsorok (prototípusok) **int** main( ) { *típusdeklarációk (lokális adatok)* *törzs* **return** 0; } *finomításdefiníciók*	*finomításdefiníciók* **Var / Const** *típusdeklarációk (a főprogram lokális adatai)* **Begin** *törzs* **End.**

Megjegyzések:

1. A C++ érzékeny a kis- és nagybetűkre, a Pascal nem (bár a következetesség mindig előnyös). A Pascalban is az azonosítókban sajátos konvenció szerint (értsd: „szabályosan") vegyesen használunk kis- és nagybetűket. (L. később.)
2. A Pascalban is létezik a finomítás-**fejsor** (szintaxisa: **Procedure**/**Function** *Finomítás-Név(formális paraméterek);* **Forward**;) és a finomítás-fejsor, ill. -definíció szétválasztása, de ritkán élünk vele. (Sokszor elkerülhetetlen rekurzívan implementált finomítások esetén.)

Típusdefiníciók

C++	Pascal
typedef struct{ típ$_1$ mező$_1$; típ$_2$ mező$_2$;…;} típ; **typedef** típE típT[db]; //a db konstans már definiálva **enum** típ{ért$_1$,ért$_2$,…}	típ=**Record** *mező$_1$:típ$_1$; mező$_2$:típ$_2$;…* **End;** típT=**Array** [1..db] **of** típE; //a db konstans már definiálva típ=(ért$_1$,ért$_2$,…);

Megjegyzések:

1. A C++ a tömböket mindig 0-tól kezdve indexeli, ezért volt elegendő a méretparaméter. A Pascalbeli tömbök indexelése tetszőleges egész-intervallummal megadható. Az egész-intervallum nemcsak 1-gyel kezdődhet (mint a fenti példában)! Sőt az **indextípus** tetszőleges ún. **diszkrét típus** lehet. Az 1..db helyett írható akár 0..db-1, sőt általában a..b. Fontos, hogy az **index-típus**nak **fordításkor meghatározott**nak kell lennie! (Diszkrét az a típus, amely véges, rendezett értékhalmazzal rendelkezik, és minden értékének –a legnagyobb kivételével– egyértelmű a rákövetkezője, illetve a –legkisebb kivételével– egyértelmű a megelőzője van.)
2. A **felsorolási típus**ban előforduló értékek nem egyezhetnek meg semmilyen más azonosítóval (pl. számkonstansok vagy más felsorolási típus konstansainak azonosítójával).
3. A **felsorolási típus**beli értékekről szabad (de nem muszáj) tudni, hogy a felsorolás sorrendjében értékük: 0, 1, …
4. Természetesen a felhasznált típusoknak már addigra definiáltaknak kell lennie.

Elemi típusok

C++	FreePascal	Bájthossz
short int *vagy* **short** **unsigned** shortint **int** *vagy* **long int** *vagy* **long** **double** **bool** **char** **string**	**Integer** **Word** **LongInt** **Real** **Boolean** **Char** **String**	2 2 4 8 1 1 256

Megjegyzés:

Egy **`String`** típusú adat 1. jele (ha nem üres a string), akkor C++ esetében a **0** indexű, Pascalban az **1** indexű. Az indexelés a két nyelvben egyformán történik, az `s[i]` jelenti az `s i` indexű jelét.

Aritmetikai operátorok

C++	FreePascal
`x++;` `x+=y;` `x--;` `x-=y;` **+, -, *** `//számok között` **double / double** **int / int, int % int** `//természetes más egész típus esetén is`	`Inc(x);` `Inc(x,y);` `Dec(x);` `Dec(x,y);` `+, -, *` `//számok között` ` /` `//számok között` **Integer Div Integer, Integer Mod Integer** `{természetes más egész típus esetén is}`

Logikai operátorok

C++	FreePascal
< **<=** **==** **!=** **>=** **>** **&&** **||** **!**	**<** **<=** **=** **<>** **>=** **>** **and** **or** **not**

Megjegyzés:

Ha a Pascal logikai kifejezésében több relációi is van, akkor azokat **külön-külön** zárójelek közé kell zárni. Például: `(0<i)` **`and`** `(i<=N)`.

Adatdeklaráció (v. típusdeklaráció)

C++	FreePascal
típ vált; **const** *típ kons*=*kif*;	**Var** *vált*:típ; **Const** *kons*:típ=*kif*;

Megjegyzés:

A deklarációk „egyesíthetők”, ha azonos típusúak a deklarációban szereplő adatok.

Például: C++ esetén „**int** a,b;” ugyanaz, mint „**int** a; **int** b;” Pascalban: „**Var** a,b:**Integer**;” ugyanaz, mint „**Var** a:**Integer**; b:**Integer**;” vagy „**Var** a:**Integer**; **Var** b:**Integer**;”

Komment

C++	(Free)Pascal
… //a sorvégi megjegyzés … /* megjegyzés akár több soron át */	… //a sorvégi megjegyzés … (* megjegyzés akár több soron át *) … { megjegyzés akár több soron át }

Értékadás

C++	FreePascal
vált=*kif*;	*vált*:=*kif*;

Elágazás

C++	Pascal
if (*felt*) **{** *akkor-ág* **}** **else** **{** *különben-ág* **}**	**If** *felt* **then** **Begin** *akkor-ág* **End** **else** **Begin** *különben-ág* **End;**

Megjegyzés:

A *felt* egy logikai kifejezést jelöl. Az utasítás **else**-szel bevezetett része elmaradhat.

Többirányú elágazás

C++	Pascal
switch (*kif*) { **case** *ért*$_1$: *ág*$_1$; **break**; **case** *ért*$_2$: *ág*$_2$; **break**; … **default** *különben-ág* }	**Case** *kif* **of** *ért*$_1$: **Begin** *ág*$_1$; **End**; *ért*$_2$: **Begin** *ág*$_2$; **End**; … **else** **Begin** *különben-ág* **End**; **End**;

Megjegyzés:

A *kif* és az *ért*$_i$-k azonos típusúak. A **default**/**else** ág elmaradhat.

Számlálós ciklus

C++	Pascal
for (**int** *cv*=k; *k*<=*v*; ++*cv*) { *cm*; }	**Var** //*lokális deklaráció* *cv*:**Integer**; … **For** *cv*:=k **to** *v* **do** **Begin** *cm*; **End**;
for (**int** *cv*=k/*lokális*/; *k*>=*v*; --*cv*) { *cm*; }	**Var** //*lokális* *cv*:**Integer**; … **For** *cv*:=k **downto** *v* **do** **Begin** *cm*; **End**;

Elölfeltételes ciklus

C++	Pascal
while (*felt*) { *cm*; }	**While** *felt* **do** **Begin** *cm*; **End**;

Hátulfeltételes ciklus

C++	Pascal
do { *cm*; } **while** (*felt*)	**Repeat** *cm*; **Until not** *felt*;

Finomítás-definíció – függvény

C++	Pascal
típ *finomításNév*(*form.param.*) { *típusdeklarációk (lokális adatok)* *törzs* **return** *fvÉrt*; }	**Function** *finomításNév*(*form.param.*):típ; **Var** *és/vagy* **Const** *típusdeklarációk (lokális adatok)* **Begin** *törzs* *finomításNév*:=*fvÉrt*; **End**;

Finomítás-definíció – eljárás (értéknélküli függvény)

C++	Pascal
void *finomításNév*(*form.param.*) { *típusdeklarációk (lokális adatok)* *törzs* **return** ~~*fvÉrt*~~; }	**Procedure** *finomításNév*(*form.param.*)~~:típ~~; **Var** / **Const** *típusdeklarációk (lokális adatok)* **Begin** *törzs* ~~*finomításNév*:=*fvÉrt*;~~ **End**;

Finomítás-fejsor (prototípus)

C++	Pascal
típ *finomításNév*(*form.param.*); **void** *finomításNév*(*form.param.*);	**Function** *finomításNév*(*form.param.*):típ; **Forward**; **Procedure** *finomításNév*(*form.param.*); **Forward**;

I/O

C++	Pascal
//adatok a sorból: **cin >>** *vált*$_1$ **>>** *vált*$_2$ **>>** …;	//számadatok a sorból: **Read**(*vált*$_1$,*vált*$_2$,…);
//egy adat a sor elejéről a *vált*-ba: **string** ***tmp***; **cin >>** *vált*; getline(cin,tmp,'\n'); //a sor végének eldobása	//egy számadat a sor elejéről a *vált*-ba: **Readln**(*vált*); {a sor végének automatikus eldobása}
cout << *kif*;	**Write**(*kif*);
cout << *kif* **<< endl**;	**Writeln**(*kif*);
cout << setw(8) **<<** *szöveg*;	**Write**(*szöveg:8*);{összesen 8 karakternyi}
cout << setw(8) **<<** setprecision(3) **<<** *valós*;/*8 széles mezőben, 3 értékes jeggyel*/	**Write**(*valós:8:3*);{összesen 8 karakternyi, 3 tizedes jeggyel}

Megjegyzés:

1. A Pascal és a C++ inputja lényegesen eltér egymástól, különösen a nem számtípusú adatok esetén! Erre némileg a fenti táblázat fel is hívja a figyelmet.
2. Figyelem, a `setprecision(3)` az értékes jegyek számát adja meg, ami korántsem biztos, hogy a tizedeseket jelenti; így a C++ és Pascal fenti egymásmellé rendelése nem jelent pontos kódolási szabályt a két nyelv között, inkább csak ötletelés.

Fájlkezelés

Mindkét programozási nyelvre igaz, hogy a standard I/O egy speciális adathordozón (konzolon) ábrázolt szöveges fájl. (Csak szöveges fájlokról lesz szó!) Így, amit a konzul I/O-ról tudunk, az igaz a megfelelő „irányú" szöveges fájlra is. Emiatt az alábbiakban csak a specialitásokat ragozzuk:

C++	Pascal
`#include <fstream>` `//inputfájl típusdeklaráció:` `ifstream iF;` `string fN;//a fájlnevet tartalmazza` `//az fN nevű fájl megnyitása inputra:` `iF.open(fN.c_str());` `//a nyitás sikerességének ellenőrzése:` `… iF.is_open() … //bool függvény` `//szám olvasása fájlból` *`vált`*`-ba:` `iF >>` *`vált`*`;` `//karakter olvasása fájlból` *`vált`*`-ba:` `iF.get(`*`vált`*`);//char típusú a` *`vált`* `//első szó olvasása fájlból` *`vált`*`-ba:` `iF >>` *`vált`*`;` `//fájlvég-figyelés:` `… iF.eof() … //bool függvény` `//fájl lezárása:` `iF.close();`	`//inputfájl típusdeklaráció:` **`Var`** `iF:Text;` `fN:String;//a fájlnevet tartalmazza` `//az fN nevű fájl megnyitása inputra:` `Assign(iF,fN); {$i-}Reset(iF);{$i+}` `//a nyitás sikerességének ellenőrzése:` `… IOResult=0 … //Boolean kifejezés` `//szám olvasása fájlból` *`vált`*`-ba:` `Read(iF,`*`vált`*`);` `//karakter olvasása fájlból` *`vált`*`-ba:` `Read(iF,`*`vált`*`);//Char típusú a` *`vált`* `//első szó olvasása fájlból` *`vált`*`-ba:` `… ilyen művelet nincs, sajnos,` `… ez bonyolultabban oldandó meg` `//fájlvég-figyelés:` `… Eof(iF) … //Boolean függvény` `//fájl lezárása:` `Close(iF);`

`//outputfájl típusdeklaráció:` `ofstream oF;` `//az fN nevű fájl megnyitása outputra:` `oF.open(fN.c_str());` `//szám írása fájlba` *`vált`*`-ból:` `oF <<` *`vált`* `<< ' ';` `//szöveg sor írása fájlba` *`vált`*`-ból:` `oF <<` *`vált`* `<< endl;` `//fájl lezárása:` `oF.close();`	`//outputfájl típusdeklaráció:` `Var oF:Text;` `//az fN nevű fájl megnyitása outputra:` `Assign(oF,fN); Rewrite(oF);` `//szám írása fájlba` *`vált`*`-ból:` `Write(oF,`*`vált`*`,' ');` `//szöveg sor írása fájlba` *`vált`*`-ból:` `Writeln(oF,`*`vált`*`);` `//fájl lezárása:` `Close(oF);`