Mérnöki programozás 4

Szerkesztette: dr. Vass Péter Tamás

C nyelv alapelemei

A C programozási nyelven írt forráskódokat felépítő alapelemek (szintaktikai egységek), az alábbi csoportokba oszthatók:

- azonosítók,
- kulcsszavak,
- állandók (konstansok),
- megjegyzések,
- operátorok,
- egyéb elválasztók.

C nyelv alapelemei

Azonosítók

A C nyelvű forráskódok bizonyos összetevőit (pl. változók, függvények, címkék) egyedi azonosító nevekkel kell ellátni.
A kódban az azonosító névvel hivatkozhatunk az adott összetevőre.
A nevek betűkből, számokból és aláhúzásjelből állhatnak.
Ékezetes betűket nem lehet használni az azonosítókban.
Egy adott betű nagy és kis változata eltérő szimbólumnak számít.
Pl. *valtozo_nev* *Valtozo_nev* két különböző azonosítónak számít.
Azonosító csak betűvel vagy aláhúzásjellel kezdődhet.
Számmal nem kezdődhet azonosító!
Az azonosítók tetszőleges hosszúságúak lehetnek, bár a fordítóprogramtól függ, hogy mennyi karaktert vesz figyelembe.

C nyelv alapelemei

Kulcsszavak

A C nyelvben léteznek olyan szavak, amelyeknek speciális jelentése van.
Ezeket a szavakat nem lehet azonosítóként használni, mert a nyelv alapvető szintaktikai elemei.
Az ilyen szavakat nevezzük kulcsszavaknak.
A kulcsszavak mindig kisbetűvel kezdődnek és a nyelv szabványa definiálja.
Kulcsszavak közé tartoznak többek között az adatok típusát és érvényességi körét jelölő szavak (pl. int, long, float, double, char, const, extern, auto stb.), valamint a vezérlési szerkezetekhez kapcsolódó szavak (pl. if, else, do, while, for).

C nyelv alapelemei

Állandók (konstansok)

Az állandókat két nagyobb csoportba oszthatjuk:

- numerikus állandók (számok),
- karakteres állandók (karakter vagy karakterek összetartozó sorozata, másnéven karakterlánc).

A numerikus konstansoknak több fajtája létezik a számok típusa és tárolási módja szerint. A tárolás itt a számítógép memóriájában történő kódolt formájú tárolását jelenti az értéknek.
A számokat ui. kódolt formában tárolja a számítógép a memóriában a program futása során.
A szám típusától (egész, valós) és egyéb jellemzőitől (nagyság, előjel) függően különböző kódolású és memória igényű tárolási formátumok léteznek.

C nyelv alapelemei

Állandók (konstansok)

A tárolási formátumok alapján két csoportba oszthatók a numerikus állandók:

- egész állandók,
- lebegőpontos állandók.

Egész állandók

Számjegyek sorozatából állnak.
Megadhatók decimális (10-es alapú), oktális (8-as alapú) és hexadecimális (16-os alapú) számrendszerekben.
A decimális egész állandók olyan egész számok, amelyek nem 0-val kezdődnek (pl. 2017).
Az oktális egészek mindig 0-val kezdődnek (pl. 03741).

C nyelv alapelemei

Állandók (konstansok)

Egész állandók

A hexadecimális egészek 0x vagy 0X előtaggal kezdődnek (pl. 0x7e1 vagy 0X7E1).

Megjegyzés:

A 16-os számrendszer az a, b, c, d, e, f, ill. A, B, C, D, E, F betűket alkalmazza a 10, 11, 12, 13, 14 és 15 számok jelölésére.

Oktális egész átváltása decimális egésszé

$03741 = 1 \cdot 8^0 + 4 \cdot 8^1 + 7 \cdot 8^2 + 3 \cdot 8^3 = 1+4 \cdot 8+7 \cdot 64+3 \cdot 512=2017$

Hexadecimális egész átváltása decimális egésszé

$0x7e1 = 1 \cdot 16^0 + 14 \cdot 16^1 + 7 \cdot 16^2 = 1+14 \cdot 16+7 \cdot 256=2017$

Itt ki az e értékét 14-nek kellett venni a számításban.

C nyelv alapelemei

Állandók (konstansok)

Előjel nélküli egész állandók

Az ilyen állandók esetében nem kell tárolni az előjelre vonatkozó információt a memóriában.
A szám értéke után kiírt u vagy U betűvel (unsigned) jelezzük a fordítónak, hogy a szám előjelnélküli egész (pl. 2017u, 03741u, 0x7e1u).

C nyelv alapelemei

Állandók (konstansok)

Nagyobb értékű előjeles egész állandók

A memória felhasználás kímélése esetében csak azokat a nagyobb értékű egész állandókat érdemes nagyobb memória területen tárolni, melyek ténylegesen igénylik a nagyobb helyet (több bájtot a memóriában).

A kisebb értékű egész állandók tárolása ui. kevesebb bájt lefoglalásával is megoldható.
A nagyobb értékű egész állandóhoz tartozó tárolási módot a szám értéke után kiírt l vagy L (long) betűvel jelezzük a fordítónak (pl. 20578962L, 0116401222L).

Nagyobb értékű előjel nélküli egész állandók

A számhoz csatolt ul, vagy UL (unsigned long) utótag jelöli.

C nyelv alapelemei

Állandók (konstansok)

Lebegőpontos állandó

A lebegőpontos állandó egy ún. lebegőpontos számábrázolás szerint tárolt 10-es számrendszerű (decimális) valós számnak felel meg.
A lebegőpontos formátum a szám normálalakú felírásán alapul.
(további részletek: https://gyires.inf.unideb.hu/GyBITT/31/ch05s05.html)
A szám normálalakjának tárolásához szükséges adatok:

- egészrész,
- törtrész,
- a 10 kitevője (egész szám).

A C nyelvű forráskódban alkalmazhatjuk a tizedes tört szerinti megadást, vagy a normálalakú felírást. Példák:

-25.302	-2.5302e2	-2.5302E2
0.0025302	2.5302e-3	2.5302E-2

A tizedesvessző helyett pontot kell alkalmazni, és a 10 hatványkitevőjét az e, vagy E (exponent) betűkkel választjuk el a szorzótényezőtől.

C nyelv alapelemei

Állandók (konstansok)

Lebegőpontos állandó

A lebegőpontos állandókat különböző pontosságú és értéktartományú típusok szerint tárolhatjuk, ami különböző méretű memóriaterület igénybevételét is jelentheti (a pontos memóriaterület nagyságok a hardvertől és a fordítóprogramtól függenek).

A tárolás pontosság és értéktartomány szerinti típusok

- egyszeres pontosságú (float),
- kétszeres pontosságú (double)
- és nagy pontosságú (long double) lebegőpontos állandók.

Az alapértelmezett tárolási pontosság a kétszeres pontosság (double).

Ha ettől eltérő pontossággal kívánjuk tárolni az állandót, akkor egyszeres pontosság esetén az f vagy F, nagy pontosság esetén pedig az l vagy L betűket kell a szám után írni (pl. 5.785f, 5.785e-20l)

C nyelv alapelemei

Állandók (konstansok)

Karakter állandó

A karakter állandók betűk, számok és egyéb szimbólumok lehetnek. Egyszeres idézőjelek (aposztróf) között kell megadni az adott karakter szimbólumát (pl. 'c' 'D' '5' '+').

A karakter állandónak van számértéke is, ami az ASCII kódjának felel meg.

(további részletek az ASCII kódról: https://hu.wikipedia.org/wiki/ASCII)

A C nyelv nem használja az ékezetes betűk karaktereit.

Vannak speciális karakterek, mint például a szövegek formázásához használt „nem látható” (white space) karakterek, és a C nyelvben más jelentéssel bíró karakterek, amelyeket speciális módon, ún. feloldójel szekvenciával (escape sequence) lehet karakter állandóként jelölni.

C nyelv alapelemei

Állandók (konstansok)

Karakter állandó

A feloldójel szekvenciák mindig egy ún. feloldójellel (escape character) kezdődnek, ami a C nyelvben a fordított osztás jelnek (backslash character) felel meg, és utána valamilyen más karakter következik. A feloldójel felhívja a figyelmet arra, hogy a következő karaktert másképpen kell értelmezni, mint általános esetben

Néhány gyakrabban alkalmazott feloldójel szekvencia a C nyelvben:

\n	újsor (soremelés) karakter
\t	vízszintes tabulátor karakter
\v	függőleges tabulátor karakter
\a	hangjelzés karakter
\\	maga a fordított osztásjel (backslash)
\’	aposztróf
\?	kérdőjel
\”	idézőjel

C nyelv alapelemei

Állandók (konstansok)

Karakterlánc (string)

A karakterlánc idézőjelek közé zárt karakterek sorozatát jeleni. (Az angol nyelv helyesírási szabálya szerint a kezdő idézőjel is felső állású.)

A karakterláncon belül alkalmazhatók az feloldójel szekvenciák is a megjelenítés formázása, ill. speciális karakterek megjelenítése céljából. Példák:

"A beszed a munka arnyeka."

A fenti karakterlánc kiírásakor megjelenő eredmény:

A beszed a munka arnyeka.

"elso\tmasodik\tharmadik"

Itt a szavak közé vízszintes tabulátor karakterek vannak beillesztve a megfelelő feloldójel szekvenciával. A kiírásakor megjelenő eredmény:

elso masodik harmadik

C nyelv alapelemei

Állandók (konstansok)

Karakterlánc állandó (sztring konstans)

"elso\nmasodik\nharmadik\n"

Itt a szavak közé újsor karakterek vannak beillesztve a megfelelő feloldójel szekvenciával. A kiírásakor megjelenő eredmény:

elso
masodik
harmadik

Az ékezetes karaktereket a C nyelv nem támogatja.

A hosszú karakterláncokat több sorba tördelhetjük a forráskódban a fordított osztásjel segítségével anélkül, hogy a kiíratáskor több sorba kerülnének a részek(pl. *"A paraszt a szamar es a lo* \
a piacra mennek.").

Minden karakterlánc konstanst egy speciális karakterrel, a null karakterrel zár le a fordító, aminek a feloldójel szekvenciája '\0'

C nyelv alapelemei

Megjegyzések

A forráskódban olyan szöveges sorokat helyezhetünk, el amelyek magyarázatokat, és egyéb információkat tartalmaznak. Az ilyen szöveges sorokat megjegyzéseknek (comment) nevezzük.

A megjegyzéseket /* és */ karakter párok közé kell zárni. Ezzel jelezzük a fordítóprogram számára, hogy ezeket a sorokat nem kell feldolgozni.

A megjegyzések folytatólagosan több sorba is írhatók pl.

```
/* Ez egy olyan megjegyzes, amit
ket sorban helyeztem el.*/
```

C nyelv alapelemei

Operátorok

Az operátorok olyan szimbólumok, amelyek segítségével valamilyen művelet elvégzését írjuk elő (pl. +, -, *, /).

Az operátorok tárgyai az ún. operandusok, amelyek memória címekkel azonosítható részei a programnak (pl. konstans, változó).

Vannak olyan operátorok, amelyek egyetlen operandusra vonatkoznak, mások két operandus között értelmezhető műveleteket írnak elő.

Az operandusokból és az operátorokból kifejezések képezhetők.

Az operátorokat általában a szerepük alapján szokták csoportosítani.

C nyelv alapelemei

A leggyakrabban előforduló operátorok

Aritmetikai operátorok:

+	összeadás,
-	kivonás,
*	szorzás,
/	osztás,
%	maradékképzés operátora (az egészosztás maradékát adja eredményül, pl. 5 % 2 = 1)

Értékadó operátorok:		Példák:		
=	egyszerű értékadás	z = x + y;		
+=	hozzáadás és értékadás	y += x;	jelentése	y = y + x;
-=	kivonás és értékadás	y -= x;	jelentése	y = y – x;
*=	szorzás és értékadás	y *= x;	jelentése	y = y * x;
/=	osztás és értékadás	y /= x;	jelentése	y = y / x;

C nyelv alapelemei

A leggyakrabban előforduló operátorok

Összehasonlító (relációs) operátorok:

<	>	kisebb, mint	nagyobb, mint
<=	>=	kisebb vagy egyenlő	nagyobb vagy egyenlő
==		(azonosan) egyenlő	
!=		nem egyenlő	

Az összehasonlító operátorokat a vezérlési struktúrák feltételeinek kifejezéseiben használjuk.
Ha egy összehasonlító kifejezés a kiértékelésekor igaznak bizonyul, akkor az értéke 1 lesz.
Ha hamis a kifejezés, akkor az értéke 0 lesz.

C nyelv alapelemei

A leggyakrabban előforduló operátorok

Logikai operátorok

&& logikai és művelet (kétoperandusú),

|| logikai vagy művelet (kétoperandusú)

! tagadás (negáció) operátora (egyoperandusú)

A logikai operátorokat szintén a feltételek létrehozásánál alkalmazzuk a vezérlési szerkezetek vezérlő utasításaiban. Az összehasonlító operátorokkal kombinálva bonyolult feltételes kifejezések alkothatók.

C nyelv alapelemei

A leggyakrabban előforduló operátorok

A logikai operátorok logikai igaz (1) és hamis (0) értékkel rendelkező operandusokra vonatkozhatnak.

A logikai ÉS művelet igazságtáblája

a	b	a && b
0	0	0
0	1	0
1	0	0
1	1	1

C nyelv alapelemei

A leggyakrabban előforduló operátorok

Logikai operátorok

A logikai (megengedő) VAGY művelet igazságtáblája

a	b	a||b
0	0	0
0	1	1
1	0	1
1	1	1

A logikai NEM művelet igazságtáblája

a	!a
0	1
1	0

C nyelv alapelemei

A leggyakrabban előforduló operátorok

Léptető operátorok

Egyoperandusú operátorok.

++ a változó értékét eggyel növeli (increment)

-- a változó értékét eggyel csökkenti (decrement)

Alkalmazhatjuk az operandus előtt (prefix) vagy után (postfix). A kétféle alkalmazás között akkor jelentkezik különbség, ha a léptetés egy másik műveletbe van beágyazva.

Előrevetett alkalmazásakor a változó értéke először megváltozik, majd ezzel az új értékkel vesz részt a következő művelet eredményének kialakításában.

A hátravetett alkalmazáskor viszont a változó a régi értékével vesz részt a külső művelet eredményének kialakításában, majd a változó értéke eggyel növekszik vagy csökken.

C nyelv alapelemei

A leggyakrabban előforduló operátorok

Léptető operátorok

Példa hátravetetett növelő operátor hatására:

```
a = 0;
b = 0;
if (a || b++)
    printf(”a vagy b igaz”);
else
    printf(”a vagy b hamis”);
printf((”b erteke: %d”, b);
```

A C nyelvű kódrészlet eredménye a képernyőre kiíratva:

a vagy b hamis

b erteke 1

C nyelv alapelemei

A leggyakrabban előforduló operátorok

Léptető operátorok

Példa előrevetett növelő operátor hatására:

```
a = 0;
b = 0;
if (a || ++b)
    printf(”a vagy b igaz”);
else
    printf(”a vagy b hamis”);
printf((”b erteke: %d”, b);
```

A C nyelvű kódrészlet eredménye a képernyőre kiíratva:

a vagy b igaz
b erteke 1

C nyelv alapelemei

Operátorok precedenciája

Egy összetettebb C nyelvi kifejezésben többféle operátort alkalmazunk.
Az operátorokhoz kapcsolódó műveletek nem mindegyike egyenrangú.
Egyesek végrehajtása elsőbbséget élvez a többiekhez képest, függetlenül attól, hogy a kifejezésen belül hol helyezkedik el.
Az operátorokra vonatkozó ún. precedencia (elsőbbségi) szabályok határozzák meg, hogy az egyes műveletek melyik szinten helyezkednek el a végrehajtási sorrendben.

C nyelv alapelemei

Operátorok asszociativitása

A precedencia szabályok szerint azonos hierarchia szinten lévő műveletek végrehajtási sorrendjét, az operátoraiknak az adott kifejezésben elfoglalt helyzetük határozza meg.

A sorrendiség iránya az egyes szinteken jobbról balra, más szinteken balról jobbra értelmezett.

A kiértékelésnek az irányát nevezzük asszociativitásnak (csoportosítás).

A C nyelvi operátorok teljes körére vonatkozó precedencia és asszociativitás szabályai a szakkönyvekben és a Web-en megtalálható.

Pl. Brian W. Kernigham, Dennis M Ritchie: A C programozási nyelv

C nyelv alapelemei

Az előzőekben bemutatott operátorok precedenciája és asszociativitása

A műveletek precedenciája (elsőbbsége) felülről lefelé csökken

!	++	--	- (előjel op.)		jobbról balra
*	/	%			balról jobbra
+	- (kivonás op.)				balról jobbra
<	>	<=	>=		balról jobbra
==	!=				balról jobbra
&&					balról jobbra
||					balról jobbra
=	+=	-=	*=	/=	jobbról balra

A zárójelpárok helyes alkalmazásával megkerülhetjük a fenti sorrendiséget! A zárójelpáron belüli kifejezés önálló egységet képez kiértékelési szempontból.

C nyelv alapelemei

Egyéb elválasztók

A C nyelvben egyéb szintaktikai szerepet betöltő, gyakran használt szimbólumok.

( ) kifejezésekben a műveletek precedenciáját változtatja meg, függvények esetében a paraméterlistáját határolja,

[ ] tömbök (azonos típusú adatok sorozata) méretének megadásakor, és a tömb egyes elemeire hivatkozáskor (indexelés)használjuk,

{ } függvények, és vezérlési szerkezetek utasítás blokkjainak lehatárolására szolgálnak,

, a függvények paraméterlistáin belül az egyes paraméterek elválasztására szolgál,

; az aktuális utasítást lezárja,

az előfordítónak szóló utasítás (direktíva) kezdő szimbóluma

A C nyelv adattípusai

A C nyelv (hasonlóan más nyelvekhez) különböző adattípusokat határoz meg. Ezekhez a típusokhoz eltérő memóriaterület foglalás, kódolási mód és műveletek tartoznak.

A memória foglalást igénylő elemekhez (pl. változók, konstansok, függvények) rendelt azonosító neveket és a hozzájuk tartozó adattípust mindig deklarálnunk kell a forráskódban.

A deklarációt még az előtt meg kell tennünk, hogy az adott elemet a forráskódon belül használnánk.

A deklaráció megadja a szükséges információkat a fordító számára, azonban még nem eredményez tényleges helyfoglalást a memóriában.

A helyfoglalás a definícióval történik meg. Minden memória foglalást igénylő elemet egyszer definiálnunk kell.

A definíció egyben deklarációnak is minősül, ha előzőleg még nem volt deklarálva az adott elem a programban.

Egy elemet többször is lehet deklarálni egy több modulból álló programban, de csak ugyanolyan módon (ugyanazon azonosítóval és típussal).

Definíciója azonban minden memóriafoglalást igénylő elemnek csak egy van az egész programon belül (akár egy modulból, akár több modulból áll a program).

A C nyelv adattípusai

A C nyelv leggyakrabban alkalmazott alaptípusaihoz tartozó típuselőírások

char	karakter szimbólumként értelmezett adat típusa (1 bájt memória foglalással jár),
int	egész számként értelmezett adat típusa,
float	egyszeres pontosságú lebegőpontos formátumban értelmezett szám típusa,
double	kétszeres pontosságú lebegőpontos formátumban értelmezett szám típusa.

Ezeken kívül létezik még három alaptípus: **enum** (*felsorolt típus*), **struct** (*struktúra típus*) és **union**.
Speciális típusnév a **void** ami a típus hiányát jelzi (*üres típus*) és speciális esetekben használatos (pl. kimeneti érték nélküli függvénynél).

A C nyelv adattípusai

A C nyelv leggyakrabban alkalmazott alaptípusai

A típuselőírásokhoz típusmódosítók is kapcsolhatók, amelyekkel az alaptípusok különböző változatai alakíthatók ki.

signed char	előjeles karakter típus
unsigned char	előjelnélküli karakter típus
long int	nagyméretű egész típus (memória helyfoglalás szempontjából nagy)
short int	kisméretű egész típus
unsigned int	előjelnélküli egész típus
unsigned long int	előjelnélküli nagyméretű egész típus
unsigned short int	előjelnélküli kisméretű egész típus
long double	nagypontosságú lebegőpontos szám típus

Az egyes típusokhoz és változataikhoz tartozó memóriafoglalási méretek a hardvertől és a C fordító programtól függenek.

A C nyelv adattípusai

Egyszerű változók definiálása

A változó egy adatok tárolására szolgáló memória terület, melyet azonosítóval (névvel) látunk el. A változóhoz tartozó memóriaterületen tárolt adat értéke megváltozhat a program futása során.

A változó definiálását még azelőtt el kell végeznünk a programban, mielőtt először használnánk a változót valamilyen kifejezésben, vagy valamilyen adat tárolására.

A változó definiálása egyben deklarációnak is minősül, és ezen a ponton a programban már memóriahelyfoglalás történik a változó számára.

Példák:

```
int elso;
int elso, masodik, harmadik;
```

A C nyelv adattípusai

Egyszerű változók definiálása

A változó definiálásakor tehát meg kell adni a típusát, majd az azonosítóját (nevét). A definíciót pontosvessző zárja le.
Több azonos típusú változó egyszerre is definiálható vessző karakterekkel elválasztva.
A definícióval együtt kezdőérték is adható a változónak (inicializáció). Példák:

```
float szigma=2.895;
float szigma, tau=-1.98, kappa;
```

A kezdeti értékadás természetesen a változó definiálása után is megtehető a forráskód további részében.
Tilos olyan változót használni egy kifejezésben, amely nem kapott előzetesen értéket akár inicializáció, akár egy másik kifejezés kiértékelése útján!
Az ilyen változó értéke meghatározatlan.

A C nyelv adattípusai

Példa az alaptípusokhoz tartozó memória foglalási méretek kiíratásához

```
#include <stdio.h>
/*a sizeof operátor az argumentumában megadott változó által
lefoglalt memóriaterület méretét adja meg bájtokban*/

main( ){
    char a;
    int b;
    float c;
    double d;

    printf("A char tipus altal foglalt memoria bajtokban: %d\n", sizeof(a));
    printf("Az int tipus altal foglalt memoria bajtokban: %d\n", sizeof(b));
    printf("A float tipus altal foglalt memoria bajtokban: %d\n", sizeof(c));
    printf("A double tipus altal foglalt memoria bajtokban: %d", sizeof(d));
}
```

Konstansok használata

A konstansok használatának két leginkább elterjedt módja:

- const típusminősítővel definiált változó formájában,
- az előfordító számára megadott szimbolikus konstans formájában.

A **const** típusminősítőt a változó definíciójában kell alkalmazni, és a változót kötelező értékkel együtt megadni. Példa:

```
const int x = 10;
```

Az ilyen módon definiált változó értékét nem lehet közvetlen módon megváltoztatni a programban.

Az előfordítónak szánt részben (a main függvény definíciója előtt) ún. szimbolikus konstansokat definiálhatunk a #define direktíva segítségével. Példa:

```
#define PI 3.14159
```

Ennek eredménye az lesz, hogy az előfordítási folyamat során az előfordító program a PI minden egyes előfordulási pontján behelyettesíti a forráskódba a szimbolikus konstans értékét .

Konstansok használata

Példa a konstansok kétféle használatára

```
#include <stdio.h>
#define PI 3.14159265

main( ){
        float alfa;
        const int c = 180;
        printf("Adjon meg egy szoget 0 es 360 fok között: ");
        scanf("%f", &alfa);
        if (alfa<0 || alfa>360)
                printf("Helytelen adat!");
        else{
                alfa = alfa * PI / c;
                printf("A szog erteke radianban: %f", alfa);
        }
}
```

C nyelvű program szerkezete

A C nyelv az ún. funkció-orientált (függvény-orientált) programozási módszertant támogatja.
Ennek lényege, hogy az összetettebb problémák megoldásához szükséges részfeladatok kezelése önálló számítási egységek, un. függvények formájában valósítható meg. Egy függvény valamilyen részfeladat megoldására szolgáló algoritmus programkódja, és egy számítási szolgáltatást biztosít más függvények és a főprogram számára
A függvényeket definiálnunk kell. A definiált függvények szolgáltatásait a függvények hívásával érhetjük el a főprogramban. Maga a főprogram is lényegében egy speciális függvény, a main függvény definíciója.
Az egyszerűbb C nyelvű programok egyetlen (.c kiterjesztésű) fájlban tárolt forráskóddal rendelkeznek. Az ilyen C programokat nevezzük egyetlen modulból álló programnak.
Léteznek több forrásfájlban tárolt, ún. több modulból álló programok is. De ezekben is csak egyetlen egy főprogram található, ami a main függvény definíciója. A többi része a forráskódnak főként deklarációkat és a főprogramban meghívott más függvények definícióit tartalmazza.

C nyelvű program szerkezete

Az egy modulból álló C nyelvű program forráskódjának általános felépítése:

1. az előfordítónak szóló utasítások (direktívák)
2. globális változók deklarációja és a főprogramon kívül definiált függvények deklarációi
3. a főprogram (main) definíciója
   változók és konstansok definíciói a főprogramban
   utasítások, vezérlési szerkezetek, függvényhívások
4. a főprogramban hívott, (nem könyvtári) függvények definíciói.

Megjegyzés:
Az un. könyvtári függvényeket a programozási környezet biztosítja. Ezek kódját már megírták és függvénykönyvtárakban (library) vannak tárolva. Nagyon sok hasznos szolgáltatást biztosítanak, amiket a főprogramunkban kihasználhatunk a megfelelő könyvtári függvény hívásával.

C nyelvű program szerkezete

Példa az egy modulból álló C nyelvű program forráskódjának általános felépítésére:

```
#include <stdio.h>      /*előfordítónak szóló direktíva, beépíti a főprogramban használt
                         könyvtári függvények  deklarációit tartalmazó fejlécfájlt*/

float sum(float, float); /*a modulban definiált függvény deklarációja*/
float avg();             /*a modulban definiált függvény deklarációja*/
float z;                 /*egy globális változó deklarációja, itt még nincs memóriafoglalás*/

main(){
   float x, y;    /*lokális változók definíciói, egyben deklaráció is, de memóriafoglalással jár*/
   printf("Adjon meg ket szamot szokozzel elvalasztva!\n"); /*könyvtári függvény hívása*/
   scanf("%f%f", &x, &y);                                   /*könyvtári függvény hívása*/
   z = sum(x, y);          /*a modulban definiált függvény hívása, egyben értékadás a globálisan
              deklarált változónak, ami már memóriafoglalást eredményez a változó számára*/
   printf("A ket szam atlaga: %f", avg()); /*könyvtári függvény hívása, a második paraméterben
                                            kerül hívásra egy modulban definiált függvény*/
}

float sum(float a, float b){          /*függvény definíciója*/
              float r;                /*lokális változó definiálása, memória foglalással jár*/
              r = a + b;
              return r;               /*függvény visszatérési értékét előíró utasítás*/
}
float avg(){                          /*függvény definíciója*/
              return z/2;
}
```