2. GAIA

DATU TIPOAK ETA ERAGIKETAK

1 C LENGOAIAREN SINTAXIA

C lengoaiaren hiztegian 6 motatako osagaiak daude: **konstanteak, identifikadoreak, hitz klabeak, operadoreak, komentarioak eta bereizleak**. Ikus dezagun osagai bakoitza zehaztasunez.

1.1 Aldagaiak

Aldagai bat programa batetan eragituko diren datuak gordetzeko erabiltzen den memoria espazioa da. Aldagai bat izen batekin identifikatzen da, programan zeharko aginduetan guk erabili nahi dugun memoria aldeari erreferentzia egiteko. Aldagai baten edukia programan zehar aldatu daiteke.

1.2 Konstanteak

Aldagaiak programan zehar bere balioa aldatu dezakete, edo baita programa berearen egikaritze ezberdinetan. Aldagaiez gain, programa batek konstanteak ere erabili ditzake, hau, beti berdinak diren balioak. Ohiko adibide bat π zenbakiarena da, 3.14.1592654 balioa duena. Balio hau, zifra esanguratsu gehiago edo gutxiagorekin, programa bateko sententziatan hainbat aldiz agertu daiteke.

C-n mota ezberdinetako konstanteak daude:

1. ***Konstante numerikoak***. Balio numerikoak dira, osoak ala koma higikorradunak.

Adibideak: $0.1*10^4$ 278 57.629

2. ***Karaktere konstanteak***. Komilla simpleen artean sartutako edozein karaktere bakarra, karaktere konstante bat legez hartzen da C-n.

Adibideak: ‘a’ ‘Y ‘)’ ‘+’

3. ***Karaktere kateak***. Komilla artean sartutako karaktere alfanumerikoen multzoa ere konstante mota bat da C lengoaian.

Adibideak: "espazioa" "Hau karaktere kate bat da "

4. ***Konstante sinbolikoak***. Konstante sinbolikoak, aldagaiak bezala, izen bat (identifikadorea) daukate. Baina EZIN DUTE bere balioa aldatu programan zehar. C-n bai ***#define*** klausula, bai ***const*** hitz klabearekin definitu daitezke.

Adibideak:

```
#define PI = 3.14
const float numE;
```

1.3 Identifikadoreak

Identifikadore bat funtzio batera edo memoria alde batera (aldagaia) ereferentzia egiten duen izena da. Lengoaia bakoitzak bere arau propioak ditu funtzio eta aldagaientzat izena aukeratzeko. ANSI C-n arau honek hurrengoak dira:

1. Identifikadore bat letra (a-tik z-ra, maiuskula eta minuskuletan) eta digitoen (0-tik 9-ra) sekuentzia batez osatzen da.

2. ***Azpimarra*** edo ***underscore*** (_) karakterea beste letra bat bezala hartzen da.

3. Identifikadore bat ezin du ez espazio hutsik ezta ez aipatutako karaktererik eduki, adibidez (*,;.:-+, etc.).

4. Identifikadore baten lehenengo karakterea beti letra bat edo (_) izan behar da, hau da, ezin du zenbaki bat izan.

5. Letra maiuskulak eta minuskulak bereizten dira. Horrela, **Balioa** eta **BALIOA** identifikadore ezberdinak dira.

6. ANSI C-k 31 karaktereko luzeera duten identifikadoreak definitzean baimentzen du.

Hauek dira identifikadore egokien adibideak:

denbora, distantzia1, A_kasua, PI, argiaren_abiadura

Alderantziz, hurrengo izenak ez dira egokiak (**¿Zergatik?**)

1_balioa, denbora-osoa, dolar$, %amaiera

Orokorrean, gomendagarria da zein aldagai tipo edo funtzio adierazten duten begi bistaz jakitean ahalbidetzen duten ***izenak*** **aukeratzea.** Honek asko errazten du programazioaren lana eta, batez ere, programen mantentze eta zuzenketa. Egia da izen luzeak idazteko lan gehiago ematen dutela, baina orokorrean errentagarria da eragozpen txiki hori hartzea.

1.4 C-ren hitz klabeak

C-en, beste edozein lengoaian bezala, erabiltzaileak identifikadore legez (aldagai edo funtzio izenak) erabili ezin dituen hitz klabe (keyword) multzo bat dago. Hitz klabe hauek ordenagailuari agindu zehatz bat adierazteko balio dute (konparaketa bat ebaluatzetik, aldagai baten tipoa definitzera) eta konpiladorearentzan esangura berezia dute.
C lengoiaia oso laburra da, beste lengoaia batzuk baino hitz klabe gutxiago ditu. Jarraian ANSI C-ren 32 hitz klabeko zerrenda aurkezten da, bere esangura geroago azalduko da xehetasun gehiagorekin (konpiladore batzuk berezko hitz klabe batzuk gehitu ditzakete)

auto	double	int	struct	break	else	long
case	enum	register	typedef	char	extern	
union	const	float	short	unsigned	continue	
signed	void	default	goto	sizeof	volatile	do
static	while	switch	for	if	return	

Hitz klabeak ezin ditugu identifikadore legez erabili

1.5 Operadoreak

Operadoreak programan agertzen diren adagai edota konstanteekin eragiketak adierazten duten zeinu bereziak dira (batzutan bi karaktereko multzoak).

C lengoaia operadore motatan bereziki aberatsa da:

- **Aritmetikoak** + - * / %
- **Esleipenekoak** =
- **Erlazionalak** == < > <= >= !=
- ***Logikoak*** && || !

1.6 Bereizleak

Bereizleak zuriune, tabuladore eta lerro berriko karaktere bat edo gehiagoz osaturik daude. Bere zeregina konpiladorea iturri programa bere oinarrizko osagaietan deskonposatzen laguntzea da. Komenigarria da zuiruneak tartekatzea nahiz eta ezinbestekoak ez izan, programen irakurgarritasuna hobetzeko.

1.7 Komentarioak

C lengoaiak programadoreak beraien programaren kodea gordetzen duten iturri fitxategietan komentarioak sartzea ahalbidetzen du. Komentarioen helburua programa nola dagoen egindari buruzko azalpen edo argipen moduan jokatzea da, horrela, programa horren eraikuntzan parte hartu ez duen edonork ulertu ahal izango du edo baita programadoreak berak ere, denbora tarte bat igaro ondoren. Komentarioak bereziki baliogarriak (eta arriskutsuak) dira ere, programa irakasleak zuzendu behar duen azterketa baten atala denean. Konpiladoreak komentarioak ez ditu kontuan hartzen bere zereginean.

(**/***) karaktereak programaren kodean tartekatutako komentario bat hasteko erabiltzen dira; komentarioak (***/**) karaktereekin amaitzen dute. Ezin daiteke komentario bat beste baten barruan sartu. Konpiladoreak inoiz ez du kontutan hartzen hasierako (/*) eta amaierako(*/) sinbolen bitartean sartuta dagoen edozein textu. Adibidez:

```
aldagai_1 = aldagai_2;
 /* lerro honetan aldagai_2-ren edukia aldagai_1-era
 esleitzen diogu */
```

Komentarioak ere, C lengoaiaren berezko beste osagai batzuen banatzaile moduan jokatu dezakete.
Akats iturri sarri bat, lehenago zabaldu den komentario bat iztea ahaztea da.

2 FUNTSEZKO DATU TIPOAK C-N

C, beste edozein programazio lengoaia bat bezala, mota ezberdineko datuekin lan egiteko aukera dauka: karaktere alfanumerikoz, zenbaki osoz, zenbaki errealez zati oso eta dezimalekin, eta abarekin osaturiko testua da. Gainera, honetariko zenbait datu tipo, zifra kopuru ezberdina (heina edo/eta zehaztasuna), bakarrik positiboak edo positiboak eta negatiboak izateko aukera, eta abar onartzen dute. Gai honetan, C lengoaiak onartzen dituen funtsezko datu tipoak ikusiko dira, eta baita beste datu tipo batzuk, funtzezkoetatik eratorriak.
Hurrengo taulak, C lengoaiaren funtsezko datu tipoak biltzen ditu.

TIPO	NOMBRE	EJEMPLOS
Caracter	***char***	‘a’ ‘3’ ‘#’
Entero	***int***	3 -8 572 -247
Real	***float***	12,57 -27,9 63,0

int hitzak zenbaki osoa dela adierazten du, ***float*** berriz, zenbaki erreal batera igortzen du. Zenbaki mota biak positiboak edo negatiboak izan daitezke.

char hitzak karaktere bat erreferentziaten du (letra maiuskula edo minuskula bat, karaktere bereziak, ...). Aipatu behar da **'3'** (komilla bakunekin) karaktere bat dela, **3** (komilla gabe) berriz, zenbaki osoa da. C-en, karaktereak zenbaki osoen antzeko tratamendua dute, ordena erlazio bat egonez eta zenbakien berezko eragiketak erabiltzea posible izanez.

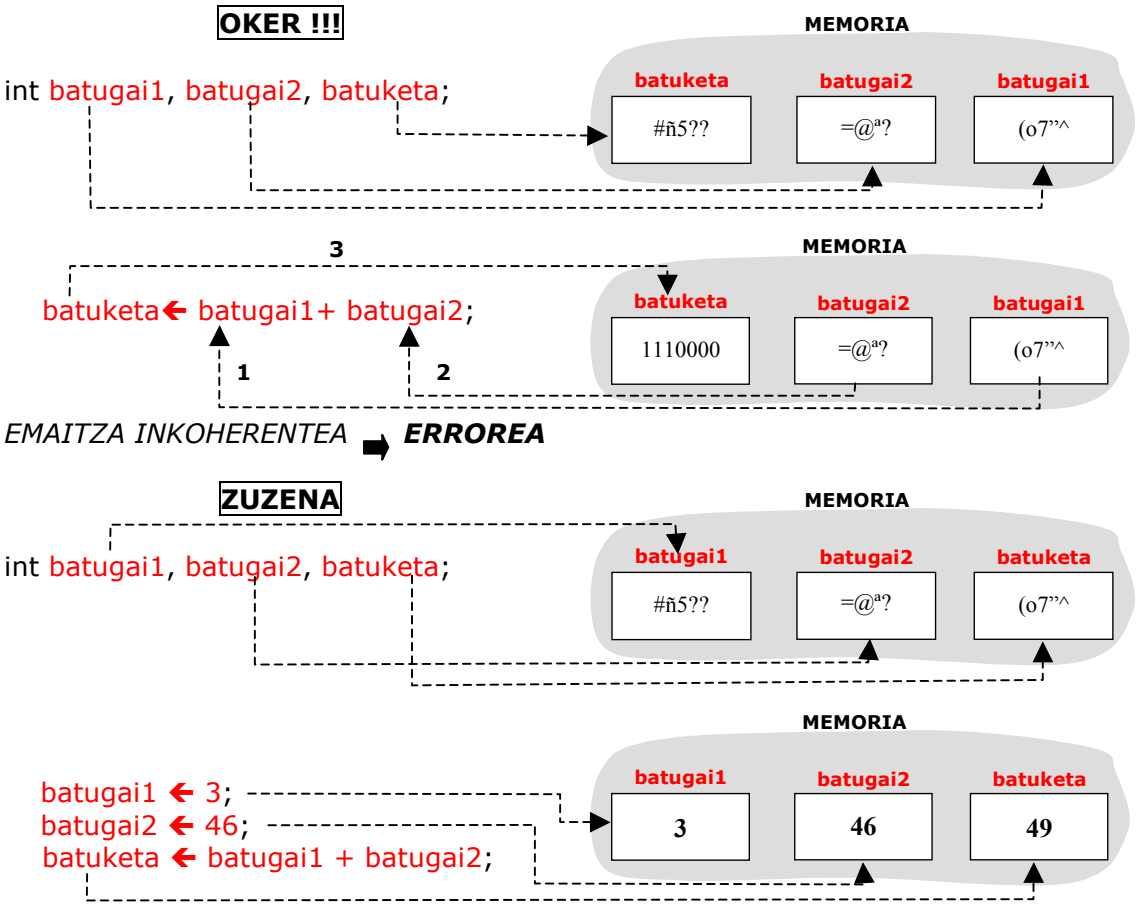
'a' < 'b' < ... 'z' 'f' + 'j' eragiketa posiblea da

***C-n* erabiliko diren aldagai guztiak deklaratzea beharrezkoa da, programan zehar edukiko duten datu tipoa adieraziz.**

Deklaratu gabeko aldagai bat akats mezu bat sorrarazten du konpilaketan.

Aldagai bat deklaratzen denean, deklarazioan adierazitako datu tipoaren arabera memoria erreserbatzen zaio. Aldagai deklarazioaren ostean, ezin da bere balioa jakin, konpiladoreak esleitu dion memoria zatiaren edukia izango delako. Izan daiteke eduki hau *zabor informatikoa* izendatzen dena izatea, beste helburu baterako memoria alde horretan lehenago gordetako balioak. Horregatik, eta gure programetak deklaratutako aldagaien eduki egokia bermatzeko helburuarekin, eragiketan erabili baino lehen egokiro inizializatzea ezinbestekoa da.

Adibideak:



Ondoren funtsezko datu tipo bakoitzeko datu bat nola den eta nola gordetzen den C-n azalduko da.

2.1 Karaktereak (char tipoa)

Karaktere (**char** tipoa) aldagaiak karaktere bakar bat gordetzen dute eta memoria ***byte*** (8 bit) betetan sartzen dira. Bit batetan bi balio gorde daitezke (0 eta 1); bi bitekin $2^2 = 4$ balio gorde daitezke (00, 01, 10 , 11 bitarrean; 0, 1, 2, 3 hamartarrean). 8 bitekin $2^8 = 256$ balio ezberdin gorde ahal izango dira (normalean 0 eta 255 bitartean; konpiladore batzuk –128 eta 127 bitarteko balioak gordetzen dituzte).
Karaktere tipoko aldagaien deklarazioa honako forma eduki dezake:

```
char izena;
char izena1, izena 2, izena 3;
```

Sententzia bakar batetan aldagai bat baino gehiago deklaratu daiteke. Deklarazioan aldagaia inizializatu daiteke baita. Adibidez, **letra** karaktere aldagaia definitu eta **'a'** balioa esleitzeko, hurrengoa idatzi daiteke:

```
char letra = 'a';
```

Hemendik aurrera, letra aldagaia definitua geratzen da, **a** letrari dagokion balioarekin. Gogoratu **letra** aldagaia inizializatzeko erabili den **'a'** balioa karaktere konstante bat dela.

Barrutik, konpiladoreak byte bakar bat erreserbatzen du **letra** aldagairentzat, bere edukia zenbaki oso bat bezala gordez, ASCII kodean **a** letrari dagokiona. Kode hau karaktere estandar bakoitzarentzat zenbakidun adierazpena ematen duen kodifikazio bat da. ASCII kode zabalago bat ere badago, karaktere bereziak eta bestelako lurraldeen alfabetoen berezko karaktereak biltzen dituena, adibidez, tildedun bokalak eta ñ letra gaztelerarako. ASCII kodeak ondoz-ondoko zenbakiak esleiten ditu alfabetikoki ordenatutako letra maiuskula eta minuskulentzat. Honek izenak alfabetikoki ordenatzeko eragiketak nabarmenki errazten ditu.

letra aldagaiaren adibidera bueltatuz, nahi denean bere balioa aldatua izan daiteke beste balio bat esleiten dion sententzia baten bitartez, adibidez:

```
letra = 'z';
```

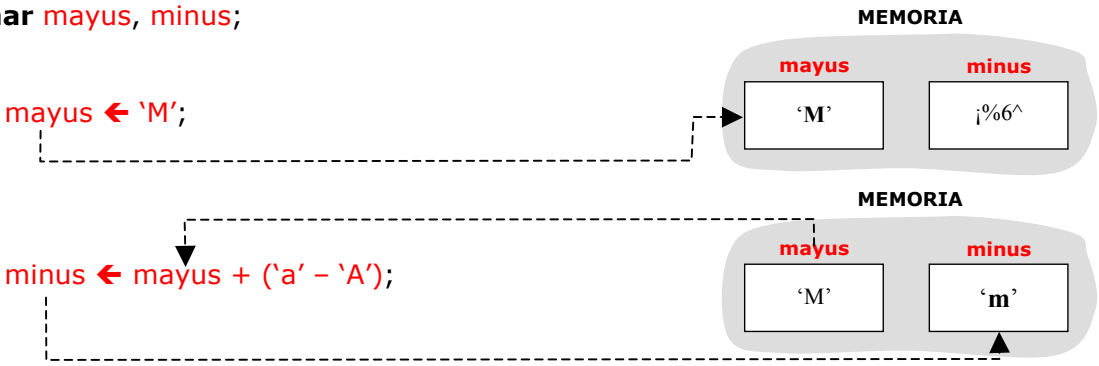
char aldagai bat ere erabili daiteke beste charaldagai bati balio bat emateko:

```
char karakterea;
karakterea = letra; // Orain karakterea 'z' balio du
```

char motatako aldagai bat zenbaki oso bat denez (0 eta 255 bitartean), bere edukia zenbaki oso bat erabiltzen den moduan erabili daiteke, horregatik, honelako eragiketak baimendutak daude:

```
letra = letra + 1;
```

Adibide honetan, **letra**-ren edukia **'a'** bazen, bat batzean, **'b'** izatera pasatzen da.



Adibide hau oso interesgarria da: letra maiuskula eta minuskulen arteko diferentzia numerikoa beti berdina denez (ASCII kodearen arabera), sententzia honek letra maiuskula bat dagokion letra minuskulara pasatzen du, aipatutako diferentzia numerikoa batuz.
Amaitzeko, gogoratu **char** tipoko aldagaiak zenbaki osoak bezalakoak direla. Aurrerago ikusiko dugu nola karaktere moduan edo zenbaki moduan idatzi ditzakegula, idazketa funtziorako deian erabiltzen den formato konbertsioaren arabera.

2.2 Zenbaki osoak (int tipoa)

Normalean, **int** aldagai bat 2 bytetan gordetzen da (16 bit), nahiz ta konpiladore batzuk 4 byte (32 bit) erabili. ANSI C-k ez du hau guztiz normalizatu eta ezberdintasunak daude konpiladore ezberdinen artean.
16 bitekin 2^{16} = 65536 zenbaki oso ezberdin kodifikatu daitezke, 0tik 65535ra zeinu gabeko aldagaientzat, eta –32768tik 32767ra zeinudun aldagaientzat (positiboak ala negatiboak izan daitezkenak), azken hau balio lehenetsia izanez.
Aldagai oso bat (int tipoa) honela deklaratu ala deklaratu eta inizializatzen da:

```
int zenbakia;
int nota = 10;
```

zenbakia eta **nota** aldagaiak –32768 eta 32767 artean egon ahal izango dira. Exekuzio denboran **int** aldagai bati baimendutako heinatik kanpo geratzen den balio bat esleiten zaionean (**overflow** edo gehiegizko balioko egoera), akats bat gertatzen da. Programak ez du erabiltzailea zirkunstantzia honetaz abisatzen, baina akatsa azaltzen da aurreikusi ezin den emaitza ondorioz eraginez.

2.3 Zenbaki osoak (long tipoa)

Hein handiagoko zenbaki osoak erabiltzeko aukera dago, bere deklarazioan **long** tipokoa bezala adierazten bada.:

```
long zenbaki_handia;
```

long zenbaki oso baten heina asko aldatu daiteke konputagailu edo konpiladorearen araber, baina normalean 4 byte (32 bit) erabiltzen dira haiek gorrdetzeko, ondorioz, 2^{32} = 4.294.967.296 zenbaki oso ezberdin adierazi daitezke. Zeinudun zenbakiak erabiltzen badira, -2.147.483.648 eta 2.147.483.647 arteko zenbakia adierazi ahal izango dira.

2.4 Zenbaki errealak (float tipoa)

Aplikazio askotan aldagai errealak behar dira, *zati osoa eta zatikiarra* duten magnitudeak adierazteko gai direnak. Aldagai hauek ere **koma higikorradunak** ere deitzen dira. Normalean, base hamartarrean eta notazio zientifikoan, aldagai hauek **mantisa** (0.1 baino

handiagoa edo berdina eta 1.0 baino txikiagoa), eta **exponentearen** (10-en berredura) bitartez adierazten dira. Zenbakia lortzeko, mantisa bider exponentea egin behar da. Adibidez, π zenbakia $0.3141592654 * 10^1$ legez adierazten da.
Bai mantisa, bai exponentea positiboak ala negatiboak izan daitezke.

Ordenagailuak base bitarrean egiten dute lan. Horregaitik **float** tipoko zenbaki bat 4 bytetan gordetzen da, 24 bit mantisarako erabiliz (1 zeinurako eta 23 baliorako) eta 8 bit exponenterako (1 zeinurako eta 7 baliorako). interesgarria da honela zein motako koma hikorradun zenbakiak adierazi daitezkeen ikustea. Kasu honetan, **heina** eta **zehaztasunaren** artean ezberdindu behar dugu. Zehaztasunak zein zifra kopuruarekin adierazten den mantisa erreferentzia egiten du: 23 bitekin adierazi daitekeen zenbakirik handiena 2^{23} = 8.388.608 da, hau da 6 zifratako zenbaki guztiak adierazi daitekez eta 7 zifratako gehienak (baina ez guztiak, adibidez, 9.213.456 zenbakia ezin daiteke 23 bitekin adierazi). Horregaitik esaten da **float** tipoko aldagaiak 6 eta 7 bitarteko zehaztasuna daukatela.

Base bitarreko mantisarekin biderkatu behar den biko exponente edo berretzaileari buruz, 7 bitekin adierazi daitekeen zenbakirik handiena 127 da. **Heina** edo errangoa berreduraren bitartez definituko da, $2^{127} = 1.7014 * 10^{38}$, hau da, honela adierazi daitekeen zenbakirik altuena. Zenbakirik txikiena balio absolutoan $2^{-128} = 2.9385 * 10^{-39}$ izango da. **float** aldagaiak honela deklaratzen dira:

```
float zenbaki_erreala;
```

float aldagaiak deklarazioaren momentuan hasieratu daitezke, **int** aldagaien antzera.

```
float zenbaki _erreala =3.14;
```

2.5 Zenbaki errealak (double tipoa)

Float motako aldagaiek oso heina eta, batez ere, zehaztasun mugatua daukate, kalkulu zientifiko eta tekniko gehienentzat, guztiz eznahikoak. Arazo hau **double** tipoarekin konpontzen da, honek 8 byte (64 bit) erabiltzen ditu aldagai bat gordetzeko. 53 bit erabiltzen dira mantisarako (1 zeinurako eta 52 baliorako) eta 11 exponenterako (1 zeinurako eta 10 baliorako). Zehaztasuna kasu honetan 2^{52} = 4.503.599.627.370.496 da, hau da 15 eta 16 bitarteko zifra dezimal adierazten ditu. Heinari dagokionez, 10 biteko berredurarekin adierazi daitekeen zenbakirik handiena $2^{1024} = 1.7977 * 10^{308}$ ingurukoa izango da.

Double motako aldagaiak, aurrekoak bezala deklaratzen dira:

```
double erreal_handia;
```

Azkenez, aldagai bat **long double** bezala deklaratzeko aukera dago, baina ANSI Ck ez du double aldagai baten heina eta zehaztasuna baino gehiago bermatzen. Konpiladore eta ordenagailu motaren menpe egongo da. Aldagai hauek honela deklaratzen dira:

```
long double erreal_handi_handia;
```

baina bere heina eta zehaztasuna ez dago normalizatuta. Microsoften konpiladoreak PC-entzat 10 byte erabiltzen dituzte (64 bit mantisarako eta 16 exponenterako).

2.6 Tipo konbertsio inplizito eta explizitoak (casting)

Eragiketa batetan mota ezberdineko aldagaiak nahastatzen direnean, Ck tipo konbertsio implizitoak egiten ditu, eragiketa burutzea eta bere eragigaiek aurrez ezarriko emaitza emotea posible egiten. Horrela, adibidez, bi aldagai batu ahal izateko, biak tipo berberakoak

izatea beharrezkoa da. Bata *int* bat bada eta bestea *float* bat, lehenengoa float tipora aldatzen da (hau da, maila txikienako tipoa duen aldagaia, maila handienakora aldatzen da), eragiketa burutu baino lehen. Tipo konbertsio automatiko eta implizito honetan, programadoreak ez du perterik hartzen, baina bere arauak ezagutu behar ditu, maila txikienako aldagaia barrutik aldatzen delako, bestearen mailara moldatzeko.
Horrela, bi motatako konstante adota aldagaiak agertzen direnean adierazpen berean eta operadore baten bitartez erlazionatuta, konpiladoreak bi eragigaiak dau tipo berdinera bihurtzen ditu heinen arabera. Hauek txikienatik handienera, honela sailkatzen dira:

long double > *double* > *float* > *long* > *int* > *char*

Beste konbertsio implizito mota bat aldagi bati espresio edo adierazpen baten emaitza esleiten zaionean gertatzen da, emaitza hori aldagaiaren tipora konbertitzen delako (kasu honetan, expresioaren maila baino txikiagoa izan daiteke, informazioa galdu daiteke eta arriskutsua izan daiteke).
Cz baita tipo konbertsio explizitoak egiteko aukera dago (**casting** izenekoak, ingeles literaturan). Horretarako, konbertitu nahi den konstante, aldagai edo adierazpenaren aurretik guk nahi dugun tipoa jarri besterik ez dugu egin behar, parentesi artean sartuta. Hurrengo adibidean,

```
k = (int) 1.7 + (int) masa;
```

masa aldagaia int tipora konbertitzen da, **1.7** konstantea bezala (*double* tipokoa dena).

3 ERAGIKETAK

C lengoaiak programa batek erabiltzen dituen datuekin mota ezberdinetako eragiketak egitea ahalbidetzen du.

Operadore bat, balio bat edo bat baino gehiagokin aritzen den karaktere edo karaktere multzo bat da, **eragiketa** jakin bat egiteko eta **emaitza** jakin bat lortzeko. Operadoreen adibide tipikoak batuketa (+), kenketa (-), biderketa (*), eta abarrenak dira. Operadoreak batarrak, bitarrak ala n-tarrak izan daitezke, zenbat eragigaien gainean aritzen diren arabera.
Cz mota exberdinetako operadore anitz daude, eta jarraian ikusiko dira.

3.1 Operadore aritmetikoak

Operadore aritmetikoak ulertzeko eta erabiltzeko errezenak dira. Guztiak operadore bitarrak dira.
Cz hurrengo bost operadoreak erabiltzen dira:

- Batuketa: **+**
- Kenketa: **-**
- Biderketa: *****
- Zatiketa: **/**
- Hondarra: **%**

Operadore guzti hauek konstante, aldagai eta adierazpenei aplikatu ahal dira. Emaitza bi eragigaiei dagokien eragiketa aplikatzeaz lortzen dena da.

Azalpen gehiago behar duen operadore bakarra **hondarra %** da. Egia esan, bere izen osoa **zatiketa osoaren hondarra** da. Operadore hau **int** tipoko konstante, aldagai eta adierazpenei bakarrik aplikatzen zaie. Behin hau argituta, bere esangura begibistakoa da:

23%4 = 3, 23/4 egitearen hondarra 3 delako.

a%b zero bada, ***a b***ren multiploa da

Adierazpen bat, operadore ezberdinez erlazionatutako aldagai eta konstanteen –eta baita beste adierazpen errezago batzuen- multzo bat da. Operadore aritmetikoak agertzen diren adierazpen adibide bat x aldagaiaren bigarren graduko hurrengo polinomioa da:

```
5.0 + 3.0*x - x*x/2.0
```

Expresio edo adierazpenak **parentesiak** (...) izan ditzake bere terminoak elkartzeko. parentesiak beste parentesien barruan agertu daitezke. Parentesien esangura adierazpen matematikoena da, geroago ikusiko diren ezaugarri garrantzitsu batzuekin. Batzutan, zuriuneen erabilera adierazpen barruan, irakurgarritasuna hobetzen du.

3.2 Esleipen operadorea

Esleipen operadorea aldagai bati, adierazpen baten emaitza edo beste aldagai baten balio ematen dio, hau da, aldagaiari dagokion memoria aldean balio hori ezartzen du.
Esleipenaren operadorea **berdintasunaren** ikurrarekin (=) adierazten da, eta ez da berdintasun matematikoarekin nahasi behar. Bere itxura orokorra hauxe da:

aldagai_izena **=** adierazpena;

Bere funtzionamendu hurrengoa da: adierazpena ebaluatzen da eta emaitza **aldagai_izena**n uzten da, aurretik hor zegoen beste edozein balio ordezkatuz. Operadore honen erabilera posible bat hurrengoa da:

```
aldagaia = aldagaia + 1;
```

Ikuspuntu matematikotik, adibide honek zentzugabekoa da (0 = 1 !!!), baina zentzua badauka pentsatzen badugu esleipen operadorea ordezkapen bat adierazten duela; hain zuzen, memorian dagoen aldagaiaren balioa hartzen da, unitatea gehitzen zaio eta emaitza berriro aldagaiari dagokion memoria aldean jartzen da, aurretik zegoen balioa ordezkatuz. Emaitza aldagaiaren balioa unitate batean handitzea da.
Horrela, aldagai bera (=) operadorearen bi aldeetara agertu daiteke.

Esleipen operadoearen ezkerretara ezin da inoiz adierazpen bat agertu:
beti aldagai baten izena egon behar da.

Ez da zuzena, ondorioz, horrelako gauza bat idaztea:

```
a + b = c; /* OKERRA */
```

Esleipen zuzenen adibideak:

```
balioa = 8;
gehiketa = 2;
emaitza = gehiketa + balioa;
distantzia = distantzia + 1;
heina = heina /2.0
x = x * (3.0 * y - 1.0)
```

3.3 Operadore erlazionalak

Edozein programazio lengoaia baten ezinbesteko ezaugarri bat, **alternatibak kontuan hartzea** da, hau da, modu batean edo bestean aritzea baldintza batzuk betetzen diren ala ez begiratuz. **Operadore erlazionalak** baldintza hoiek betetzen diren ala ez egiaztatzea ahalbidetzen dute.
Lengoaia naturalean, hitz edo forma ezberdinak daude baldintza zehatz bat betetzen den ala ez erakusteko. Inglelez hitz hoiek (***yes***, ***no***), (***on***, ***off***), (***true***, ***false***), eta abar dira. Baldintza bat betetzen bada, emaitza true da (egia); alderantziz, emaitza false da (gezurra).
Operadore erlazionalak Cz hurrengoak dira:

```
Berdina: ==
Baino txikiagoa: <
Baino handiagoa: >
Txikiagoa edo berdina: <=
Handiagoa edo berdina >=
Ezberdina: !=
```

Operadore erlazional guztiak operadore N **bitarrak** dira (bi eragigai dituzte), eta bere itxura orokorra hurrengo a da:

expresio1 **op** *expresio2*

op hauetako operadore bat dela: ==, <, >, <=, >=, !=

Operadore hauen funtzionamendua hurrengoa da: **expresio1** eta **expresio2** ebaluatzen dira, eta emaitzako balioak konparatzen dira. Operadore erlazionalak adierazitako baldintza betetzen bada, emaitza true da (egia); baldintza betetzen ez bada, false da (gezurra).
Ondoren, konstanteei aplikatutako operadoreen adibideak erakusten dira:

```
(2==1)  /* emaitza = false baldintza betetzen ez delako */
(3<=3)  /* emaitza = true baldintza betetzen delako */
(3<3)   /* emaitza = false baldintza betetzen ez delako */
(1!=1)  /* emaitza = false baldintza betetzen ez delako */
```

3.4 Operadore logikoak

Operadore logikoak operadore erlazionalen emaitzak konbinatzea ahalbidetzen duten operadoreak dira, baldintza bat baino gehiago betetzen diren, bata ala bestea betetzen den, eta abar egiaztatuz.
C lengoaiak bi operadore logiko ditu: **ETA** operadorea (&&) eta **EDO** operadorea (||). Ingelesez, operadore hauek **and** eta **or** deitzen dira. Bere itxura hurrangoa izaten da:

```
expresio1 || expresio2
expresio1 && expresio2
```

&& operadoreak true bihurtzen du bai expresio1 bai expresio2 egia direnean, eta false alderantzizko kasuan, hau da, expresio bat edo biak gezurra direnean; beste alde batetik, || operadoreak true bihurtzen du, behintzt expresio bat egia denean.
Garrantzitsua da kontutan hartzea C konpiladoreak adierazpen hauen exekuzioa optimizatzen ahalegintzen direla, eta honek, batzutan, nahi ez diren efektuak izan litezke. Adibidez: && operadorearen emaitza egia izateko, bi expresioak egiazkoak izan behar dira; expresio1 ebaluatzen bada eta gezurrezkoa bada, ez da beharrezkoa expresio2 ebaluatzea, izan ere, ez da ebaluatzen. Antzeko zerbait || operadorearekin gertatzen da: expresio1 egiazkoa bada, orduan ez da expresio2 ebaluatu behar.

&& eta || operadoreak beraien artean konbinatu daitezke (beharbada parentesi artean elkartuta), interpretazio zailago bateko kodea eskuratuz.

Adibidez:

```
(2==1) || (-1==-1) /* emaitza true da */
(2==2) && (3==-1) /* emaitza false da*/
((2==2) && (3==3)) || (4==0) /* emaitza true da */
((6==6) || (8==0)) && ((5==5) && (3==2)) /* emaitza false da */
```

3.5 Operadore unario batzuk

Orain arte ikusitako operadoreez gain, C lengoaiak beste operadore batzuk ditu, eta hoien artean jarraian erakusten diren operadore unarioak daude:

- **Kenketa** operadorea (–).
  Operadore honen efektua adierazpen batetan aldagaiaren edo ondoren datorren adierazpenaren zeinua aldatzea da. Orokorrean itxura hau dauka:

  - *expresioa*

- **Gehi** operadorea (+).
  Operadore honek aurretik ikusitako (-) operadorearen konplementoa izatea dauka helburuz. Aldagai edo adierazpen baten aurrean ipini daiteke, operadore unario baten moduan, baina egia esan, ez du ezer egiten.

  + *expresioa*

- **Ezeztapen logiko** operadorea (!).

  Operadore honek false itzultzen du true balio bati aplikatuz gero, eta true itzultzen du false balio bati aplikatuz gero. Bere itxura:

  ! *expresioa*