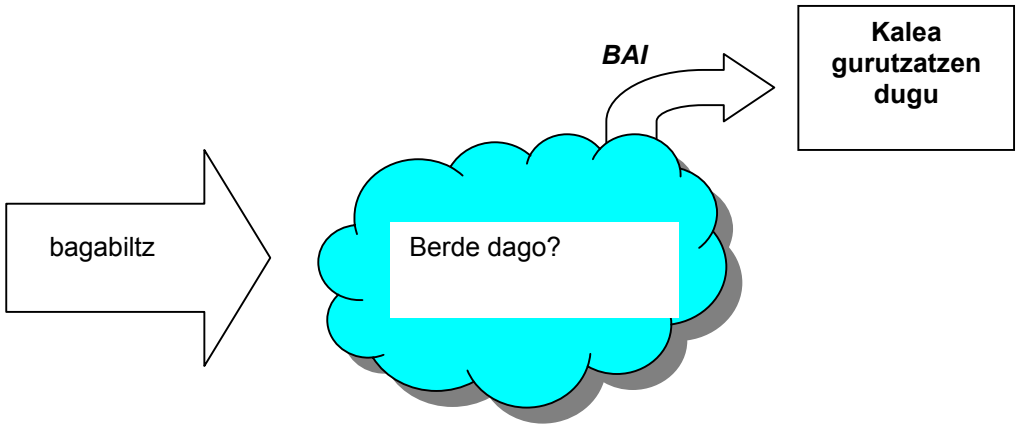
3. GAIA
BALDINTZAZKO EGITURAK

1 SARRERA

Orain arte ikusi dugu programa batek aginduak bata bestearen ondoren (sekuentzialki) betetzen dituela, pantalla bitartez balioak erakutsi ditzakegu, datuak eskatu, eragiketa matemático errezak burutu,...
Baina, bizitzan bezala, dena ez da prozedura finko batera mugatzen, beti ordena berean, ezta beti agindu beretara ere. Baldintzazko egituren prestamenak, programa ahaltsuago, moldagarri eta dinamikoagoak eraikitzea ahalbidetzen dute.
Baina programazio eraginkor eta ulergarri on bat programaren diseinu on bat implikatzen du, beraz, adierazpenek nola "arrazonatzen" duten ezagutzea funtsezkoa da.

1.1 Dena sekuentzialki funtzionatzen du?

Demagun kaletik gabiltzala eta semaforodun oinezko pasagune batera hurbiltzen garela. Zer egiten dugu? Erantzuna: galdetu. Semaforoaren kolorea berdea den galdetu genezake. Horrela balitz, kalea gurutzatuko genuke.
Beste era batera esanda:



2 BALDINTZAK

Aurreko egoera aurkezteko momentuan, galdetu nahi dugun baldintza adierazi behar dugu. Lortzeko modu posible bat pseudokodaren bitartez edo pseudo-lengoaia naturalaren bitzartez da.:

Semaforoa berdea ***BADA***
ORDUAN *kalea gurutzatzen dugu*

Horrela, *semaforo_kolorea* izeneko char tipoko aldagai baten pentsatu ahal izando genuke, gorriarentzat 'g' eta berdearentzat 'b' balioak hartzen dituena.
Galdetzeko erreminta falta zaigu. Modu naturalean (eta erderaz) "**si** semáforo_kolorea **es verde**" esango bagenu, C lengoaiaz honala adieraziko litzateke:

```
if ( semaforo_kolorea == 'b') {
      printf("orain kalea gurutzatzen dugu");
}
```

Ikus dezagun:

- ***if*** *"ba"* baten baliokidea da (baldintzazko konjuntzioa). Ohar gaitezen minúsculas idazten dela eta ez beste era batean (C lengoaia oso sentikorra da maiuskula eta minuskuletara, bien artean ezberdintzen du, beraz, konpiladorearentzat 'C' eta 'c' karaktere ezberdinak dira).
- **==** berdintasunaren operadorea da, lehenengo operadorea eta bigarrena berdinak diren egiaztatzeko.
- Kasu honetan *semáforo_kolorea* 'b' karakterea den egiaztatuko du.
- Galdera parentesi artean idazten da.
- Aginduak giltzen artean idazten dira, zenbat sententziatan aurkeztuta dauden kontutan hartu gabe.

Baina azaltzen zaigun galdera, semaforo_kolorea == 'b' adierazpena benetan esan nahi duena jakitea da, hau da, nola interpretatzen den egiatan.
Adierazpen bat ebaluatzen denean, logika bitarreko balio batetara itzultzen da: egia ala gezurra. C lengoiaiaz, benetan gertatzen dena zera da, edozein adierazpen, ebaluatua izatean, balio bat hartzen duela. Balio hori 0 bada, orduan *gezurra* bezala ulertzen da; zero ez den beste edozein balio *egiaren* baliokidea da.
Adibidez, bi aldagai ditugu:

Talde1puntuak	Talde2puntuak
5	*10*

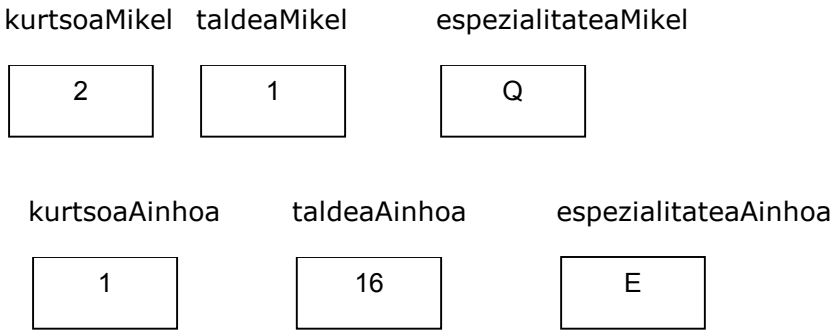
Talde1puntuak == 4 adierazpenak gezurra balioa hartzen du, hau da, 0 balio numerikoaren baliokidea. Hurrengo galdera egitean:

```
if (Talde1puntuak == 4) {
      printf("Lehenego taldeak ez ditu 10 puntu.");
}
```

Erantzuna *gezurra* da, beraz, ez da pantailatik testurik idatzi behar izango. Modu informalean, "if-ren adarratik ez dela sartzen" esaten da, edo, bestela, "ez dela if-an sartzen".
Bilatzen dena bere balioa beste balio zehatz bat ez dela bada, orduan erabili behar den operadorea *!=* da.

```
if (Talde1puntuak !=  10) {
      printf("Lehenengo taldeak ez ditu 10 puntu.");
}
```

Ariketak:
EUITIko ikasle guztien informazioa gordeta daukagu.



1.- Adierazi itzazu, if aginduen bitartez, hurrengo egiaztapenak:

a.- Mikel hirugarren kurtsoan ez dagoela.
b.- Ainhoa electrónica espezialitatekoa dela.
c.- Ainhoa eta Mikel espezialitate berekoak direla.
d.- Mikel Mekanika ('M') espezialitatekoa dela.
e.- Mikel eta Ainhoa kurtso ezberdinetan daudela.

2.- Hurrengo egoeratan erantzun zuzena adierazi ezazu.

a.- if (kurtsoaAinhoa != kurtsoaMikel)
b.- if (taldeaMikel == 2)
c.- if (kurtsoaAinhoa != taldeaMikel)
d.- if (espezialitateaAinhoa != 'M')

Zenbait operando tartean sartzen dituzten galderak ere egin ditzakegu. Horrela, bi taldeetatik zeinek dauzkan puntu gehiago jakin dezakegu:

```
if (puntuakTalde1 > puntuakTalde2) {
        printf("Lehenengo taldeak bigarrenak baino puntu gehiago dauzka");
}
```

Badaude beste operadore batzuk bi balio numerikoki ebaluatzeko:

Sinboloa	Deskribapena	Adibidea
>	Baino handiagoa	*puntuakTalde1 > 3* (lehenengo taldeak 3 puntu baino gehiago dituen egiaztatzen du)
>=	Baino handiagoa edo berdina	*puntuakTalde1 >= 3* (lehenengo taldeak 3 edo 3 puntu baino gehiago dituen egiaztatzen du)
<=	Baino txikiagoa edo berdina	*puntuakTalde1 >= 3* (lehenengo taldeak 3 edo 3 puntu baino gutxiago dituen egiaztatzen du)
<	Baino txikiagoa	*puntuakTalde1 > 3* (lehenengo taldeak 3 puntu baino gutxiago dituen egiaztatzen du)

Oharra: operadore matematikok (≥,≤, ≠) ez bezala, operadoreak C lengoaiaz bi sinbolokin idazten dira (>=, <=, *!*=).
Gainera, lengoaia matematikoak 1 < x ≤ 20 motako adierazpenak onartzen dituen bitartean, operadoreak Cz bitarrak dira, hau da, bi operando besterik ez dute onartzen.

Orduan, aurreko aderazpena bi adierazpen bitar errazagoen arabera eraiki beharko litzateke:
Ade batetik 1 < x, eta bestetik x ≤ 20, eta gainera, bi baldintzak derrigorrez betetzea eskatuz.

2.1 Logika kontua ...

Lehenago azaldu den moduan, logika bitarra adierazpenen formulazioan datza. Adierazpen hauek ebaluatuak izandakoan, egia ala gezurra moduan interpretatzen den emaitza batetan ondorioztatzen dute. Beste alde batetik, adierazpen logikoak bakunak edo konposatuak izan daitezke, hau da, termino bat edo batzuk implikatu dezakete.
Adibidez, "datorren astebururan Parisera joango naiz egun eguzkitsua bada eta bidaia ordaintzeko dirua badaukat" esaten badugu, orduan Parisera joateko bi baldintzak batera bete beharko dira. Hau da, hauxe bete behar izango da:

egun_eguzkitsua egia da **ETA** daukadan_dirua txartel_prezioa baino handiagoa da

Beste era batera:

egun_eguzkitsua == 'S' **AND** daukadan_dirua > txartel_prezioa

C lengoaiaz honela aderaziko litzateke:

egun_eguzkitsua == 'S' **&&** daukadan_dirua > txartel_prezioa

Horrela, bi terminoko edozein AND adierazpenerako, emaitza hurrengoa da:

		Bigarren terminoa	
		Egia	Gezurra
Lehenengo terminoa	Egia	***egia***	*gezurra*
	Gezurra	*gezurra*	*gezurra*

Edo berdina dena, AND adierazpen bat (zehatzago, AND operadore logiko bat) egia izateko, bere bi terminoak ere egia izan beharko dira.
Hala ere, adierazpen logiko baten barruan ere beste mota bateko erlazioa izan genezake. Ikus dezagun:

"datorren asteburuan Parisera joango naiz eguna eguzkitsua bada **edo** bidaia ordaintzeko dirua badaukat" esaten badugu, esan nahi izango du Pareisena joan ahal izango dudala, hau betetzen ba da:

egun_eguzkitsua egia da ED**O** daukadan_dirua txartelaren prezioa baino handiagoa bada

beste era batera:

```
egun_eguzkitsua == 'B' OR daukadan_dirua > txartel_prezioa
```

Cz:

```
(egun_eguzkitsua == 'B') || (daukadan_dirua > txartel_prezioa)
```

2.2 Ezeztapenaren opeadore bakuna

Operadore logiko bitarrak (AND eta OR) ikusi baditugu ere, bester operadore bakun berezi bat dago, ezetapenarena. Forma sinple batean, operadore honek adierazpen baten aurkako logikoa ezartzen duela esan genezake. Adibidez, *x>=8* (x zortzi baino handiagoa edo

berdina da) adierazpena baduakagu, adierazpen horren ezeztapena honela idazten dugu: *!(x>=8)*. Honek esan nahi du x ez dela 8 bezalakoa edo 8 baino handiagoa, edo beste era batera esanda, x zortzi baino txikiagoa da (x < 8).
x>=8 adierazpena ebaluatzean egia izango balitz, orduan aurkakoa ebaluatzean, gezurra eskuratuko genuke, eta alderantziz.
Laburpen bat eginez:

adierazpena	! adierazpena
Egia	***Gezurra***
Gezurra	***Egia***

Operadoreen bitarteko adierazpenen konposaketa eta eraldaketa

```
Athletic-ek bigarren mailara jaisten da hau gertatzen bada
!(athleticek_irabazi && errealak_galdu) ➔
                              (!athleticek_irabazi) || (!errealak_galdu)

(egia A edo B gezurra badira)

Athetic-ek Kopa irabazten du, hau gertatzen bada
!(madrilek_irabazi || bartzelonak_irabazi) ➔
                        (!madrilek_irabazi) && (!bartzelonak_irabazi)

(egia A edo B gezurra badira)

Athletic-ek punturen bat ateratzen du hu gertatzen bada
!( puntuak_athletic == 0) ➔ (puntuak_athletic != 0)

Erreala Athletic-ek baino gorago geratzen da, hau gertatzen bada
!( puntuak_athletic <= puntuak_erreala) ➔
                              (puntuak_athletic >= puntuak_erreala)
```

Praktika

Ondorengo puntu egoerarekin eta athleticek_galdu, errealak_irabazi, madrilek_galdu, bartzelonak_galdu, adierazi baldintzak hurregoa gertatzeko:

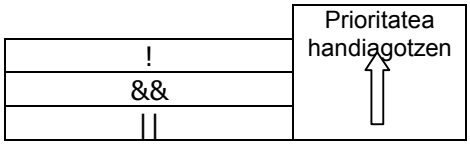
a) Athletic-ek Liga irabaztea
b) Athletic-ek lehenengoa ez geratzea

Athletic	42 puntu
Bartzelona	44 puntu
Erreala	41 puntu
Madril	40 puntu

3 ADIERAZPEN KONPLEXUEN EBALUAKETA

Aderazpen konplexuek, baldintza sinpleagoetatik aurrera eraikitzen diren egoerak deskribatzen dituzte. Ondorioz, adierazpen konplexuen ebaluaketa adierazpen sinpleen ebaluaketan deskonposatu daitekela esan dezakegu.
Adierazpen konplexuak idatzi eta ulertzeko modurik errazena, anbiguotasuna ezabatzen duten parentesiak erabiltzea da. Baina hau ez da beharrezkoa, hortaz, C lengoaiak adierazpenak ulertzeko bere modu propioa du.
Baldintza konplexu bat badaukagu eta parentesirik erabiltzen ez badugu, bi egoera hartu beharko ditugu kontutan:

1. Adierazpenak ezkerretik eskuinera ebaluatzen dira, beraien artean prioritate ezberdintasunik ez badago.
2. Beraien artean prioritaterik badago, orduan prioritate ordena kontutan hartu beharko da.



Adierazpen bi hauek:

(45 > 10) || (num > 100 + 5) || (12 = = -63)

eta

((45 > 10) || (num > 100 +5)) || (12 = = -63)

baliokideak dira.

Parentesiak ez dira beharrezkoak baldintza konplexuan operadore berdina erabiltzen bada. Operadore ezberdinak erabiltzen badira, kontuz ibili behar izango dugu prioritateekin eta ziur aski parentesiak erabili behar izango ditugu**.**

4 ELSE

Batzutan ez ditugu bakarrik agindu zehatzak bete behar egiaztatzeko baldintza batetzen bada, baizik eta hain zuzen ere, alderantzizko kasuan (betetzen ez denean).

```
if (euria_ari_da == 1) {
      printf("Euritakoa hartu");
} else {
      printf ("Eguzkitako betaurrekoak hartu");
}
```

5 AGINDU BALDINTZATUAK ETA "GILTZETATIK KANPO GERATZEN DENA"

Baldintzazko agindu bat eraikitzean, baldintza betetzen denean (orduan if barruan sartuko da) eta betetzen ez denean (orduan else barruan sartuko da) zein akzio burutuko diren adierazten ari gara. Kasu bietan, burutzeko aginduak giltzen artean mugatuta daudenak dira, hau da, giltzen artean ez dauden sententzak ez dira exekutatuko. Ikus dezagun adibide bat:

```
1     if (notaFI >=9 ) {
2           printf ("Bikain bat atera duzu");
3     } else {
4           printf ("Ez duzu bikain atera");
5     }
6     notaFI = 7;
```

Agindu hauen amaieran, hauxe edukiko dugu: pantailan bigarren testua agertzen dela, eta gainera, notaFI aldagaiaren balioa 7 dela hasierako balioa edozein izanda ere.

Ikus dezagun beste adibide bat:

```
1     if (notaFI >=9 ) {
2           printf ("Bikain atera duzu");
3           notaFI = 5;
```

```
4     } else {
5          printf (“ez duzu bikain atera”);
6     }
7     printf(“geroarte!!”);
```

Aginduen amaieran bi aukera izango ditugu: hasierako nota 9 edo gehiago bada, lehenengo testua pantailan agertu eta notaFI aldagaiaren balioa 5 izango da. Baldintza betetzen ez bada, “ez duzu bikain atera” mezua agertuko da eta notaFI-k hasierako balioa izango du. Edozein kasutan agertuko dena “geroarte!!” testua izango da.

6 BALDINTZEN SEKUENTZIA ETA ANIDATZEA

Baldintzazko aginduak hainbat modutan multzokatu daitezke: bai sekuentzian, eta bai era anidatuan ere. Jarraian idazten dira baldintza bat ebaluatu ondoren, eta ebaluaketaren emaitza edozein bada ere, beste ebaluaketa bat egitea beharrezkoa denean.
Adibidez:

```
if ( armairuAltuera >= 100) {
    altua = ‘B’;
} else {
    altua = ‘E’;
}

if ( armairuZabalera >= 200) {
    zabala = ‘B’;
} else {
    zabala = ‘E’;
}
```

Azkenean bi aldagai ditugu (altua eta zabala) bata bestearengandik aparte aldatzen direnak.

Hala ere, agindu hauen kasuan:

```
if (armairuAltuera >= 100) {
    altua= ‘B’;
    if (armairuZabalera > 1000) {
          altuaEtaOsoZabala =’B’;
    } else {
    altuaEtaOsoZabala = ‘E’;
} else {
    altua =’E’;
    altuaEtaOsoZabala = ‘E’;
}
```

altuaEtaOsoZabala aldagaia bi era ezberdinetan aldatzen dela daukagu, armairua altua den ala ez (lehenengo if) eta armairuaren zabaleraren arabera. If-ak anidatu daitezke, bai egiazko adarrean (betetzen denean) eta baita gezurrezkoan ere (betetzen ez denean).
C lengoaiaz gehienez 256 anidaketa daude onartuta, baina gu ez gara konplesutasun horretara helduko, batez ere kasu horretan programaren diseinua ez litzatekelako ez errez ezta ziur aski eraginkorra ere izango.
Beste alde batetik, baldintza multzo bat baten baino gehiagota anidatu daitekela jakinda, argi geratu behar zaigu zer den gu idazten ari garena, beraz, nahi ez den anbiguotasuna sahiesteko, else bat hurbilen dagoen fi-ra lotzen dela xedatuko dugu.
Baina kodea idazteko modu erraz bat, bai programazio orduan, bai programen arazketan ere zailtasuna xedatzeko, programaziorako estilo gida bat erabiltzea da. Estilo gidak asko aldatzen dira programadore batzuetatik beste batzuetara (bakoitza bere abantail eta desabantailekin), baina edozein kasutan, nahi ez diren akatsat xedatzen dituzte eta programen ulergarritasuna sustatzen dute.
Adibidez, nahiz eta C lengoaiak agindu guztiak lerro beren jartzea onartu, zenbait lerrotan banatzea irakurgarritasuna errazten duela egia da. Tabuladore, zuriune, parentesien erabilera ere zentzu honetan laguntza bat da.

Agindu berdinentzat, estilo ezberdinak:

```
if (kolorea ==’b’) {
      printf(“berde kolorekoa da.”);
} else {
      printf (“ez da berde kolorekoa.”);
}
```

Lerro kopuru txikiagoa. Giltzen itxiera kontrola begibistaz egiten da if-aren azpian

```
if (kolorea ==’b’)
{
      printf(“berde kolorekoa da.”);
}
else
{
      printf (“ez da berde kolorekoa.”);
}
```

Lerro gehiago. “{“ giltzaren azpian “}” agertu behar izango du.

```
if (kolorea ==’b’)
      {
      printf(“berde kolorekoa da.”);
      }
else
      {
      printf (“ez da berde kolorekoa.”);
      printf (“beste kolore batekoa da.”);
      }
```

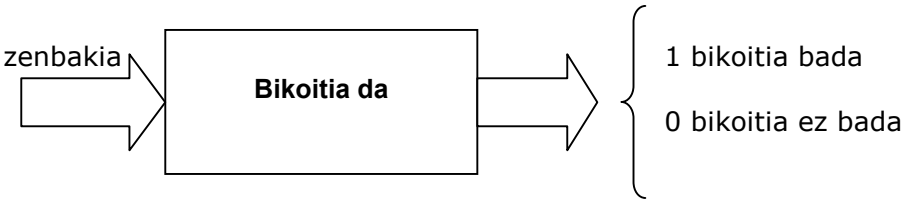
Lerro gehiago. If eta else sententzien tabulazioak baldintzako blokeen artean bereiztea ahalbidetzen du.

Ariketak:
Kanpingerako materiala prestatzen:
Adierazi ezazu mezuen bitartez, zein material eramango dugun datorren astebururan eguzkia, euria, aizea dagoen kontutan hartuz.
Euria ari bada, euritakoa eramango dugu. Gainera hotz egiten badu, berokia hartuko dugu. Baina euria ari ez bada, prakak eramando ditugu. Euria ez egiteaz aparte, eguna eguzkitsua bada, eguzkitakoa eta eguzki-krema eramango ditugu. Eguzkia badago eta haizea ere, txapela hartuko dugu.

7 BALDINTZA NUMERIKO ETA ALFABETIKOAK BAKARRIK EBALUATU DITZAKEGU?

Bistan denez, erantzuna ez da. Horrela izango balitz, programa guztiak – edozein motatakoak – adierazpen matematikoetara mugatu beharko lirateke, eta hori ez da beti posible. Besta era batera esanda, operadoreengandik gora galderak egin ditzakegu. Nola?
Demagun mekanismo edo kutxatxo beltz antzeko zerbait daukagula, guk zer nahi dugun esatean, emaitza moduan gura galderaren erantzuna ematen diguna. Kasu horretan egin dezakeguna, ematen digun emaitza numerikoki ebaluatzea da.
Adibidez, honako baldintza hau daukagu:

```
if ( bikoitia_da (zenbakia) == 1) {
      printf (“zenbakia bikoitia da”);
}
```



Funtzioak izeneko mekanismo hauek (funtzio bat betetzen dutelako, kasu honetan, bikoitia den ala ez guri esatea) ere ebaluatuak edo galdetuak izan daitezke. Nahiz eta funtzioak aurrerago ikasi, garrantzitsua da funtzio bat erabiltzean zein emaitza itzultzen digun ezagutu dezakegula jakitea. Baina ez bakarrik hori, baizik eta emaitza horretatik aurrera ere emaitza horri buruz galdetzeko aukera dugu.

8 A ERANSKINA – PRIORITATE ETA ASOZIAZIO GUZTIEN KOADROA

Aurrekotasuna	Operadorea										Asoziazioa ☐
1	**( )**	**[ ]**	**++**	**--**							☐
2	**!**	**+**	**-**	**++**	**--**	**&**					☐
4	*****	**/**	**%**	Operadore aritmetikoak: biderketa, zatiketa eta hondarra							☐
7	**<**	**<=**	**>**	**>=**	Operadore erlazionalak, baino handiagoa, baino txikiagoa,...						☐
8	**==**	**!=**	Berdina eta ezberdina operadore erlazionalak								☐
12	**&&**	ETA operadore logikoa									☐
13	**||**	EDO operadore logikoa									☐
15	**=**	***=**	**/=**	**%=**	**+=**	**-=**	**&=**	**^=**	**|=**	**<<=**	**>>=** ☐

Ohartu aurrekotasun indikadoreak ez duela ordena sekuentziala jarraitzen. Hau taulan agertzen ez diren beste operadore batzuk daudelako gertatzen da (adibidez zenbaki bitarrentzako operadoreak), baina bere prioritatea daukatela jasota geratzen da.