# 2. gaia: PROGRAMAZIOAREN METODOLOGIA

Ikasgai honen helburua ez da C programazio lengoaia ikastea, programatzen ikastea baizik. Horretarako, gai honetan programazioaren oinarrizko kontzeptuak ulertzen saiatuko gara hala nola erabili beharreko teknikak.

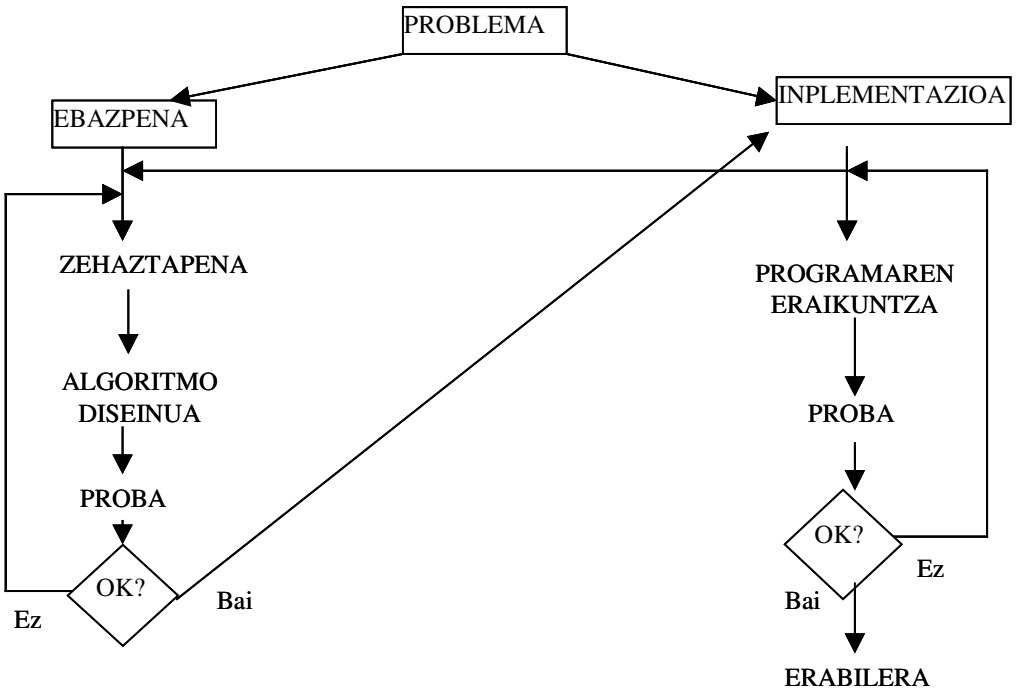
## *Definizioak*

**Programa:** Programatzaileak ordenadoreari zer burutu behar duen adierazteko ematen dizkion agindu edo sententzi multzoa.
**Programazioa:** Problema bat ebazteko pentsatutako programa informatiko baten agindu-sekuentzia erabakitzeko ekintza.
**Programazio lengoaia:** programa bat osatzeko erabiltzen diren erregela-multzoa, sinboloek eta hitz bereziek osatzen dute. Lengoaia artifiziala da, ordenadore programak adierazteko sortutako lengoaia.
**Algoritmoa**: problema bat ebazteko ekintza multzo esplizitua da.

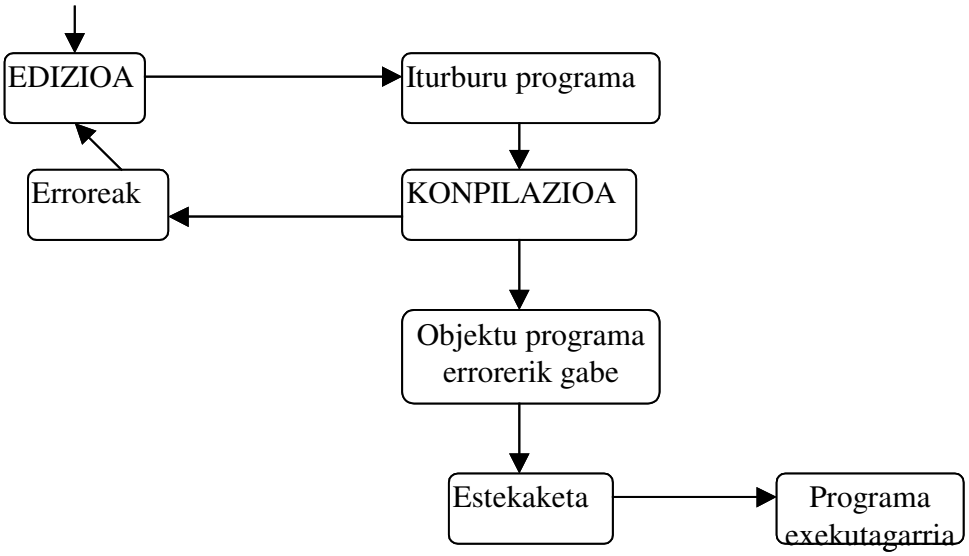
Ondoren zenbait lengoaia maila azaltzen dira haien konplexutasunaren arabera sailkatuta.

- Giza pentsakera
- Lengoaia naturala (euskara, gaztelera, …)
- Goimailako programazio lengoaia (Pascal, COBOL)
- Mihiztadura-lengoaia
- Makina-lengoaia

Programa bat idazteko orduan pauso desberdinetatik pasa behar da, jarraian adierazten da programen bizi-zikloa



Programen sorkuntza fase gehiagotan bana dezakegu:




## ALGORITMOAK

Algoritmo bat problema bat ebazteko ekintza multzo esplizitua da. Hemen programazioaren inguruan ikusten dugu kontzeptu hau baina bizitzako arlo guztietan azaltzen dira algoritmoak. Konturatu gabe ikasitako zerbait da.

Algoritmoetan datuak aldagaien bitartez maneiatzen dira eta eragiketekin moldatu. Baita konstanteak eta baldintzak agertzen dira. Algoritmoak problema ebazten duela egiaztatzeko Simulazio Taulak erabiltzen dira.

**Algoritmoen ezaugarriak**

–Zehatza izan behar du
–Pausoen arteko ordenaren adierazpena azaldu behar da
–Bi aldiz jarraituz emaitza berdina eman behar du
–Beti amaitu behar da
–Programazio lengoaiarekiko independentea izan behar du

**Adibidea: (Algoritmo literala)**

*1. Zenbakia irakurri A aldagaian*
*2. A aldagaia negatiboa bada joan 3.1 pausura*
*3. A aldagaia negatiboa ez bada joan 3.3 pausura*
*3.1 (A * -2) eragiketaren emaitza B aldagaian gorde*
*3.2 Joan 3. pausura*
*3.3 (A * 2) eragiketaren emaitza B aldagaian gorde*
*3.4 Joan 3. pausura*
*3. Pantailaz azaldu B aldagaiak duen balioa*

Beraz, problema bat ebatziko duen algoritmo bat sotzerakoan kontutan izan behar ditugu **DATU EGITURAK** (konstanteak eta aldagaiak), datuak maneiatzen dituzten **AGINDUAK** (aritmetiko eta logikoak) eta **INSTRUKZIO-SEKUENTZIA** bideratzen duten egiturak.

## ZEHAZTAPENA

Normalean, problemak horrelakoak izaten ohi dira: "sarrerako datu hauekin nola lortu emaitza". Sarrerako datuak eta emaitza erlazionatuta egoten dira. Problemen enuntziatuak hizkuntza naturalean adierazitakoak izanik era azkotan ulertuak izan daitezke ala problema guztiz zehazteko informazio falta izaten dute. Horregatik algoritmoa idatzi baino lehen beharrezkoa da problema guztiz zehaztea. Sarrerako eta irteerako datuak nolakoak diren eta nola erlazionatzen diren zehaztuko dugu (datu abstrakzioa atalean ze datu motak erabili ditzakegun zerrendatzen da).

Horretarako aurrebaldintzak eta ondorengo baldintzak ezarriko dizkiogu problemari. Zehaztapenean ondorengo lau galderei erantzuten badiegu guztiz zehaztuko dugu problema.

AURREBALDINTZA:

1-zeintzuk dira sarrerako datuak?
2-ze motako da datu bakoitza?
3-ze baldintza indibidual bete behar ditu datu bakoitzak?
4-ze erlazio dituzte sarrerako datuak?

ONDORENGO BALDINTZA:

1-zeintzuk dira irteerako datuak (emaitzak)?
2-ze motako da datu bakoitza?
3-ze baldintza indibidual bete behar ditu datu bakoitzak?
4-ze erlazio dituzte beraien artean eta sarrerako datuekin?

## DATU ABSTRAKZIOA

Behin problema zehaztuta dugula, algoritmoak sarrerako datuekin emaitza lortzeko ze lan egin behar den adieraziko du. Beraz, algoritmoak problemaren datuak logikoki maneiatzeko ALDAGAIETAN gorde beharko ditu. Honi DATU ABSTRAKZIOA deitzen diogu. Aldagaiek izen eta balio bat dute (zenbait aldagai konposatuek izen bat dute baina balio multzo bat gordetzen dute). Izenaren bitartez aldagaiaren balioa erabili eta aldatu dezakete aginduek.

**Izena:**
Aldagaiaren izena letra batekin hasten den identifikadore alfanumerikoa izango da. Adibidez: A; Kont, soldata, adina1, adina2, etab.

**Balioa:**
Aldagaiak ze balio mota gordetzen duen adierazi behar da, mota horretako balioak soilik gordeko bait ditu. Erabili ditzakegun datu motak edo egiturak honakoak dira:

Datu mota sinpleak:

- **Osoko zenbakia**: datu mota hau *osoa* hitzarekin adieraziko da. Balio adibideak: 4, -12000, 876… Aldagai-erazagupen adibidea: *osoa Kont;*
- **Zenbaki erreal**: datu mota hau *erreala* hitzarekin adieraziko da. Balio adibideak: 4.6, -12000.0, 876.987… Aldagai-erazagupen adibidea: *erreala Soldata = 1220.78;*
- **Karakterea**: alfazenbakizko karaktereak dira. Datu mota hau *karakterea* hitzarekin adieraziko da. Kakotx sinple artean adieraziko dugu balio zehatza. Balio adibideak: 'a', 'A', ' ', 'K', '4', '$'… Aldagai-erazagupen adibidea: *karakterea K = 'K';*
- **Erakuslea**: memoriako helbide bat da. Normalean aldagai bat apuntatzeko erabiltzen da. Erakusleek datu mota zehatz bati apuntatu behar diote. Erakuslea *** ikurrarekin adieraziko da, bera baino lehen apuntatzen den datu mota adierazten da eta ondoren aldagaiaren izena. Algoritmoak erakusleen bitartez apuntatzen duen datua maneiatu dezake. Aldagai-erazagupen adibidea: *erreal *erakErrealak;*

Datu mota konposatuak:

- **Taulak**: datu mota sinple bateko datu multzo bat gordetzen du. Taulek dimentsioak dituzte. Taula *[zenb]* ikurrak dituelako ezagutzen da. *zenb* zenbat elementu gordetzen dituen adierazten du. *[zenb]* ikur kopuruak dimentsio kopurua adierazten du. Aldagai-erazagupen adibidea: *osoa T[5][8];*

**AGINDUAK**:

Algoritmoetan erabili ahal ditugun eragiketak, besteak beste, hauek dira:

- **Esleipena**: = ikurraren eskerrean kokatzen dugun aldagaiak balio berri bat hartzen du, berdin ikurraren eskuinean dagoen adierazpenak sortzen duen emaitza. Beraz bi aldeak datu mota berdinekoak izan behar dute. Adibidez: *Kont = 2*4;*
- **Eragiketa matematikoak**: adierazpen matematikoak idazteko. Biderkaketa (*), kenketa (-), zatiketa (/), zatiketa osoaren hondarra (%), etab.
- **Eragiketa logikoak**: adierazpen logikoak idazteko. Edo (||), eta (&&),ez (!).
- **Sarrera/Irteerako aginduak**: datuak algoritmora "sartzeko" *IRAKURRI* agindua erabiliko da. Algoritmoaren erabiltzaileak teklatuz idazten duen balioa aldagai batetan gordetzen du aginduak. Datuak algoritmotik "ateratzeko" *IDATZI* agindua erabiliko da. Pantailaz agertuko du balio bat.
- **Bestelako aginduak**: berez aginduak ez dira baina algoritmoa non hasten den eta non bukatzen den adierazteko *HASIERA* eta *BUKAERA* etiketak erabiltzen dira.

**INSTRUKZIO-SEKUENTZIA**

Algoritmoa adierazteko bi aukera ditugu: SASI-KODEAN ala FLUXU-DIAGRAMAN idaztea. Bietan aginduak eta aldagaien erazagupena berdin idatziko ditugu. Intrukzio-sekuentzia eta kontrol-egiturak nola irudikatzen diren da bien artean aldatzen den gauza bakarra.

Instrukzioen egikaritze ordena zehaztuta dago. Aginduak bata bestearen atzetik exekutatzen dira. Zenbaitetan kontrol-egiturak daude zeintzuk baldintza logiko baten arabera exekuzio ordena aldatzen dute.

**SASI-KODEA**

Algoritmo literalek interpretazio zabala dute eta beraz gaizki ulertuak gerta daitezke. Hori ekiditeko arau batzuk ezartzen zaizkio algoritmo mota honi:

- Ordena goitik beherakoa da eta enumerazioa desagertzen da.
- Egitura pribilegiatu batzuk erabili daitezke exekuzioaren nondik norakoa aldatu dezaketenak.
- Exekuzioan jauziak egiteko algoritmo literaletan erabili daitezken galderak eta JOAN aginduak ez dira sasi-kodean baimentzen. Horren ordez, egitura pribilegiatu soilik erabili behar dira.

SASI-KODEA deritzo zeren programazio lengoaiek erabiltzen dituzten egituren antzeko kontrol egiturak erabiltzen ditu.

**Egitura pribilegiatuak edo kontrol egiturak:** *"[" eta "]" ahutazko zati bat dela adierazten du.*

**Sekuentziala:**

```
HASIERA
        agindu1;
        agindu2;
        …
        aginduN;
BUKAERA
```

**Baldintzatuak:**

i. Sinplea:

```
BADA (egi-baldintza)
        aginduak1;
BUK_BADA;
```

ii. Bikoitza:

```
BADA (egi_baldintza)
        aginduak1;
BESTELA
        aginduak2;
BUK_BADA;
```

iii. Anitza:

```
BALDIN (aldagaia)
        BADA balio1 :
                aginduak1; [IRTEN;]
        BADA balio2 :
                aginduak2; [IRTEN;]
                        …
        BADA balioN :
                aginduakN; [IRTEN;]
        [BESTELA : aginduakX;]
BUK_BALDIN;
```

**Errepikakorrak:**

i. Baldintza, Aginduak:

```
DENBITARTEAN (egi_baldintza)
        aginduak1;
BUK_DENBITARTEAN;
```

ii. Aginduak, Baldintza:

```
EGIN
        aginduak1;
DENBITARTEAN (egi_baldintza);
```

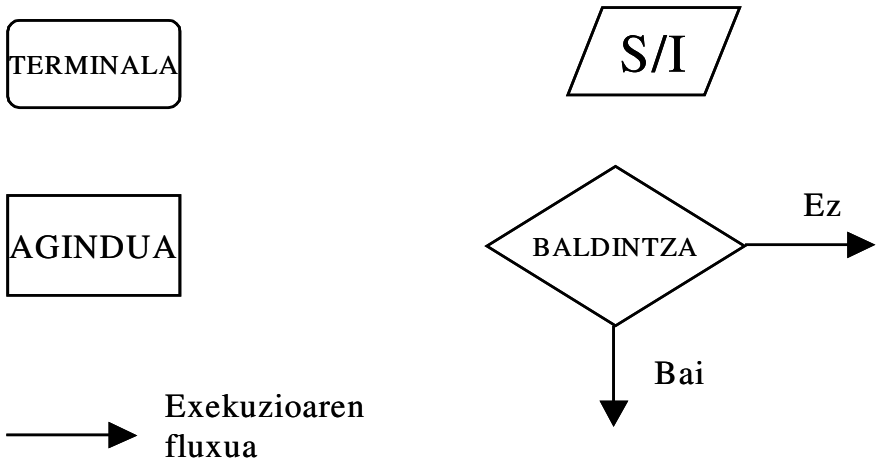
iii. Itzuli bakoitzean balioak eguneratu:

```
IZANDADIN (hasierako_balioak) DENBITARTEAN (egi_baldintza) EGUNERATU (balio_eguneraketa)
        aginduak1;
BUK_IZANDADIN;
```

## FLUXU DIAGRAMAK

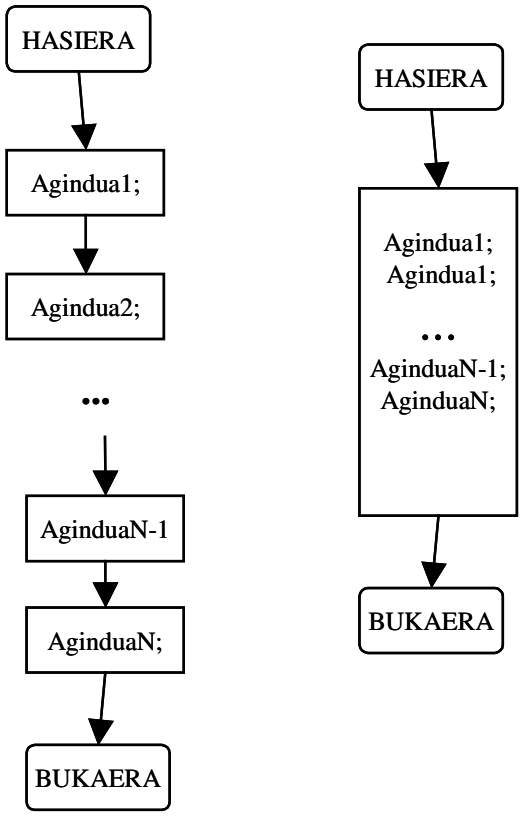
Algoritmoak sasi-kodean edota fluxu diagramen bidez adierazi daitezke. Biek berdina adierazten dute baina azken hauetan diagramak dauzkagu, askotan ulergarritasuna errazten delarik.

Fluxu diagrametan hurrengo sinboloak erabiltzen dira:



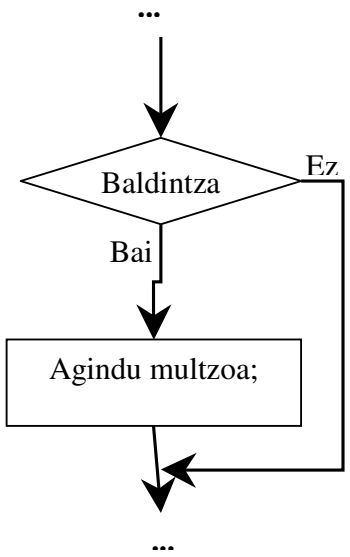

**Egitura pribilegiatuak fluxu diagrametan:**

**Sekuentziala**:

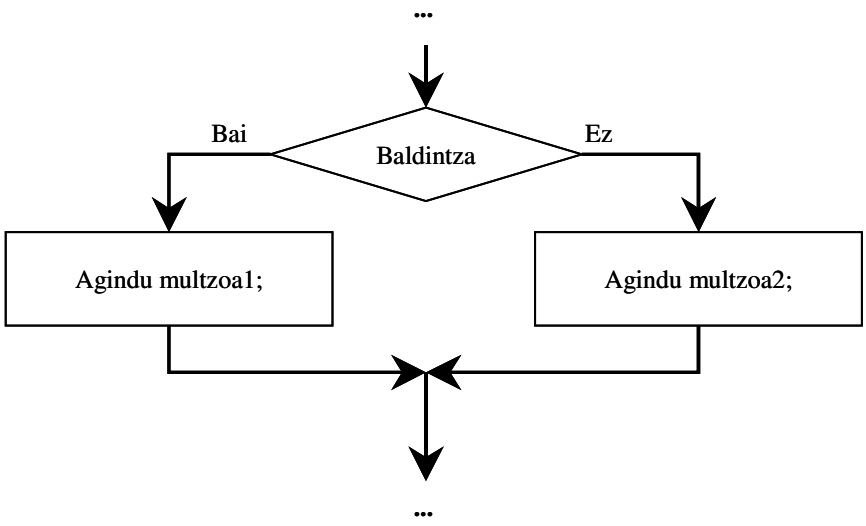


**Baldintzatuak**:

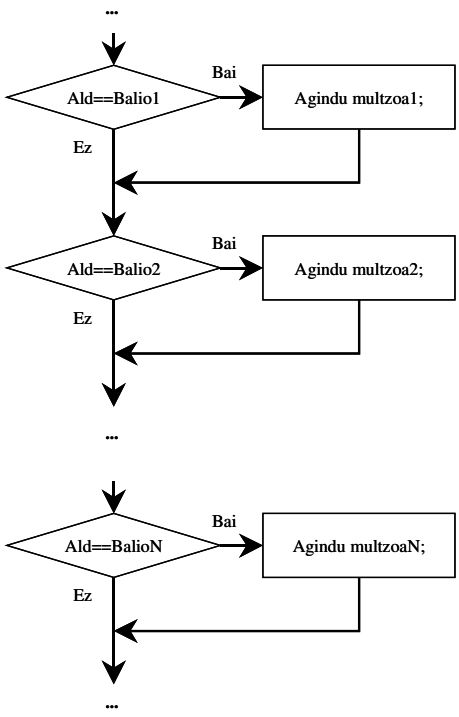
Sinplea

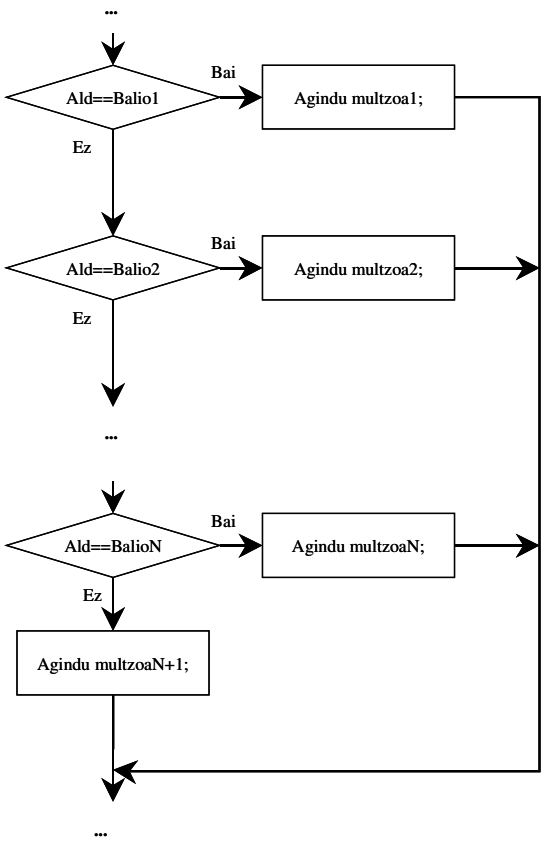



Bikoitza:



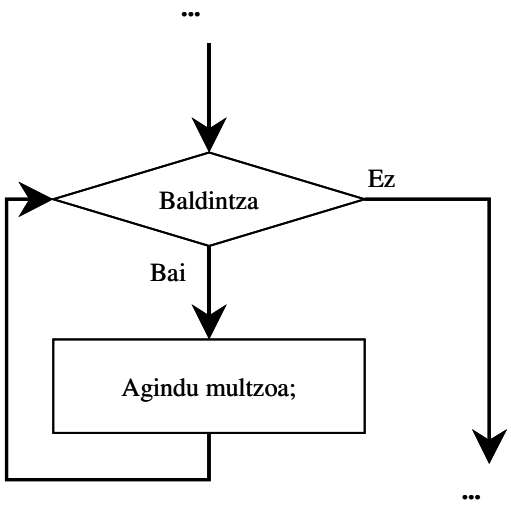
Anitza:



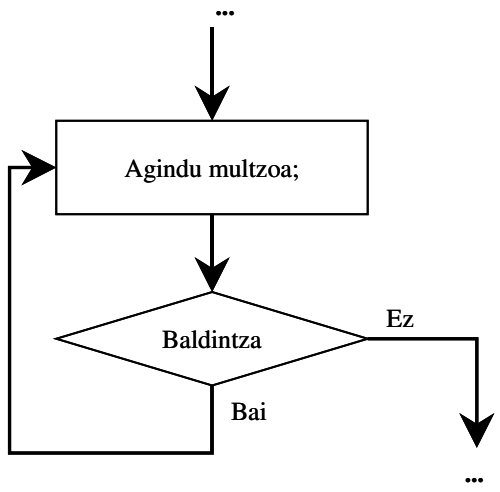




**Errepikakorrak:**

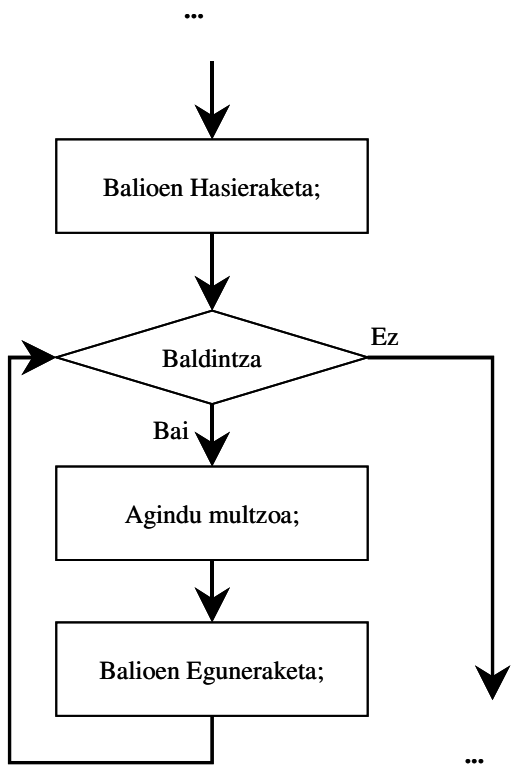
Baldintza-Agindua:



Agindua-Baldintza:




Itzuli bakoitzan balioak eguneratu:

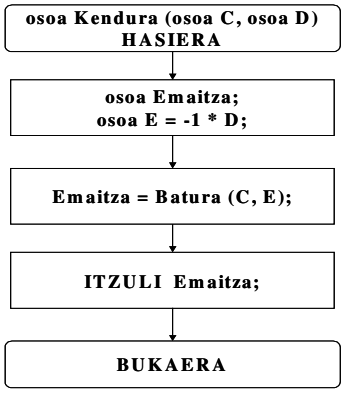


## PROGRAMAZIOA MODULARRA:

Problema handiegia bada arazoak izan ditzakegu bere osotasunean ebazteko. Hobe dala problema handiak zatika ebaztea jakina da. Programazio modularrak filosofia hori jarraitzen du.

Adibidez, bi zenbaki osoen kalkulagailu bat (batu, kendu eta biderkatu) sortu nahi bada, hobe da alde batik batura ebaztea, bestetik kenketa eta azkenik biderkaketa. Problema handi bat ebatzi beharrean hiru problema txiki ebazten ditugu. Ebazpen bakoitza modulu edo azpialgorimoa baten egongo da eta deia eginez exekutatzen da. Algoritmo nagusiak egiten dio dei sarrerako datuak pasata emaitza bat (edo gehiago) bueltatu dezan. Sarrerako datuak ez dizkio erabiltzaileak ematen baizik eta algorimo nagusiak parametroen bitartez. Azpialgoritmoak algoritmo nagusiari emaitza bueltatzen dio. Gainera azpi-aloritmoak berrerabili daitezke

```
osoa Batura (osoa A, osoa B)
HASIERA
        osoa Em;
        Em = A +B;
        ITZULI Em;
BUKAERA
```




```
osoa Biderkadura (osoa F, osoa G)
HASIERA
        Osoa Em = 0;
        DENBITARTEAN (F > 0)
                Em = Batura (Em, G);
        BUK_DENBITARTEAN
        ITZULI Em;
BUKAERA
```

```
hutsa Kalkulagailua ()
HASIERA
        osoa Auk, X, Y, Z;
        EGIN
                EGIN
                        IDATZI ("1. Batu");
                        IDATZI ("2. Kendu");
                        IDATZI ("3. Biderkatu");
                        IDATZI ("4. Irten");
                        IDATZI (" Zer egin nahi duzu?");
                        IRAKURRI Auk;
                DENBITARTEAN (Auk < 1 || Auk > 4);
                IDATZI ("Ze bi zenbaki osoekin?");
                IRAKURRI X, Y;
                BALDIN (Auk)
                        BADA 1:     Z = Batura (X, Y);
                                    IDATZI Z;
                                    IRTEN;
                        BADA 2:     Z = Kendura (X, Y);
                                    IDATZI Z;
                                    IRTEN;
                        BADA 3:     Z = Biderkadura (X, Y);
                                    IDATZI Z;
                                    IRTEN;
                        BESTELA:
                                    IDATZI "Eskerrik asko kalkulagailua erabiltzeagatik";
                BUK_BADA;
        DENBITARTEAN (Auk != 4);
BUKAERA
```