C PROGRAMAZIO-LENGOAIA (XI)

Aurrekonpiladorea eta liburutegia

Iñaki Alegria & Montse Maritxalar

C lengoaiaz programatzen dugunean, konpiladoreek zein estekatzaileek (linker izenekoek) eskaintzen dizkiguten laguntzak nola erabil ditzakegun azaltzea da kapitulu honen helburua.

Konpilazio-prozesua bi fasetan gertatzen da: lehenean (aurrekonpilazioa deitutakoan) aurrekonpiladoreari zuzendutako sententziak edo sasiaginduak tratatzen diren bitartean, bigarrenean konpiladore guztiek burutzen duten itzulpen-prozesua gertatzen da, eta ondorioz, objektu-programa lortzen da.

Estekatzaileak objektu-modulu bat edo gehiago lotzen ditu (tink), haien arteko erreferentzia gurutzatuak ebatziz eta programa exekutagarria lortuz. Lotzen diren moduluak erabiltzailearenak edota estandarrak izan daitezke, azken hauen ezaugarriak honako hauek direlarik: C liburutegian daude, aukera asko eskaintzen dute eta C-k duen ahalmenaren atal nagusietako bat da.

Aurkeztutako elementu hauek azaltzeko, ondoko hiru ataletan bananduko dugu kapitulu hau: aurrekonpiladorea, liburutegi estandarra eta erabiltzailearen liburutegiak izeneko ataletan hain zuzen.

Aurrekonpiladorea

Aurrekonpiladoreak eskaintzen dituen aukera nagusiak honako hiru hauek dira:

- Konstante parametrizatu eta makroen definizioa, #define sasiaginduaren bidez.
- Beste iturburu-modulu bat edo gehiago programan sartzea, #include sasiaginduaren bidez.
- Baldintzapeko konpilazio-zatiak ezartzea, #ifdef eta #endif sasiaginduen bidez

#define sasiagindua

Sasiagindu honen bidez konstante eta makroen definizioa egiten da seigarren kapituluan ikusi genuenez. Beraz, izen bati balio edo agindu-sekuentzia egokitzen zaio, eta ondorioz, konpiladoreak izena aurkitzean, izen horri dagokion balioa edo agindu-sekuentzia ordezkatu eta konpilatu egiten du. 1. programan adibide bat ikus daiteke:

1. programa. ***#define*** *erabiltzen deneko adibidea.*

```
#define MAX 65.535
#define ABSOL(x) ((x < 0)? - x : x)
int min abs (taula, luz)
int taula [ ], luz;
{/' taula baten balio absolutu txikiena'/
int i, txiki  = MAX;

if (ABSOL (taula [i]) < txiki)
   txiki = ABSOL (taula [i]);
return (txiki);
```

Definizioaren esanahiak moduluaren bukaeraraino irauten du #undef sasiagindua erabiltzen ez bada. Sasiagindu berri hau interesgarri gertatzen da definizioari esanahi berri bat emateko; konpiladore batzuek birdefinizioa baino lehen hala egitea eskatzen bait dute.

#include sasiagindua

Sasiagindu honen bidez iturburu-programa bat beste baten barruan sar daiteke, liburutegiaren kontzeptua iturburu-mailara hedatuz.

Sasiagindua idazteko era hau da:

#include <fitxategi izena> edo
#include "fitxategi izena"

Fitxategi izena-k sartu nahi den fitxategiak duen izena zehazten du eta .h motakoa izatea gomendatzen da; horrela bereizten bait dira izenburu (header) deituta-

ko fitxategiak. < eta > edo " erabiltzen desberdintasuna, fitxategiaren kokapenean datza. C-ren izenburu-fitxategi estandarra bada, < eta > artean jarriko da konpiladoreak ezagutzen duen katalogoan bila dezan. " artean egoteak, aldiz, erabiltzailearen izenburu-fitxategia dela adierazten du eta sistema eregileak adierazitako katalogoan bilatuko du.

Izenburu-fitxategiak liburutegiekin eta konpilazio banatuarekin batera erabiltzen dira. Funtzio batzuk modulu banatu batean konpilatzen badira estekatzailea arduratuko da moduluen arteko loturaz, baina modulu banatuan erabiltzen diren parametroak, datu--motak, aldagai globalen definizioa zein funtzioen motak beste moduluetan erazagutu behar direnez, erosoena izenburu-fitxategi batean aipaturiko definizioak sartzea da. Fitxategi hori modulu banatuaren funtzioek erabiltzen dituzten programetan sartuko da include baten bidez. 2. programan honen adibide bat ikus daiteke:

```
/' logika . h fitxategiaren edukina "/
/' eta, edo eta ala funtzioak modulu banatuan
#define TRUE 1

typedef int booi;
extern booi eta ( );
extern booi edo ( );
extern booi ala ( );
```

```
/' probalog. C programa */
#include "logika . h"
main ( )
{booi a,b,c;
a = TRUE; b = FALSE;
c = edo (a,b);
```

2. programa. #include sasiagindua nola erabili.

#ifdef, #endif sasiaginduak

Sasiagindu hauen bidez baldintzapeko konpilazioa egitea posiblea da. Konpilazio-aukera honen bidez agindu-sekuentzia bat programaren barruan sartuko da definizioa indarrean badago, ala konpiladoreak ez du kontutan hartuko, definizio hori ez badago. Definizioa kontrolatzeko #define eta #undef sasiaginduez gain konpilazio-komandoaren (cc edo msc) -D aukera erabil daiteke.

Baldintzapeko aukera hau interesgarria suertatzen da makina desberdinetarako programa bat garatzen ari garenean eta makinaren baterako tratamendu berezia egin behar denean.

3. programa. Baldintzapeko konpilazioaren adibidea.

```
#ifdef IBM
  int reg dat inp = 0x378
#endif
#ifdef LEMON
  int reg dat inp = Ox 3B8
#endif
```

Adibidez, programa batek inprimagailuaren erregistroa zuzenean maneiatu behar badu, 3. programan azaldutakoa egin daiteke makina desberdinetako helbideak kontutan hartzeko.

Liburutegi estandarra

6. kapituluan azaldu genuenez, funtzioak erabili baino lehen definitu egin behar dira, baina badago aurredefinituta dagoen funtzio-multzo bat; funtzio estandarraren multzoa, edo, gauza bera dena, liburutegi estandarreko funtzioak. Batzuk aztertu ditugu printf eta scanf bezala, baina besteak ez.

1. taulan funtzio garrantzitsuenen zerrenda agertzen da; sarrera/irteerakoak salbu, hauek hurrengo kapituluan azalduko dira eta. Funtzioen ondoan, dagokien izenburu-fitxategiaren izena zehazten da; funtzio estandarrak erabiltzeko dagokien izenburua #include batez zehaztu behar bait da.

4. programan memoria dinamikoa erabiltzen duen modulu bat azaltzen da eta 1. eskeman dagokion konpilazio- eta estekaketa-programak.

Erabiltzailearen liburutegiak

Programazio egituratuak eta softwarearen injineritzak agindu bezala, programak luzeak direnean funtziotan banatu behar dira, funtzio-multzoak osatuz, horietako bakoitza bere aldetik konpilatu eta probatuko delarik. Horrela sortutako moduluak liburutegi

```
/* lista bat memoria
 estatikoaz osatzea */
#define MAX 100
 main ( )
{  int n,i,zenb;
   int list [MAX];
   printf ("zenbat balio?:");
   scanf ("%d",&n);
   if (n>MAX) printf ("error\n");

       scanf ("%d",& list [ i]);
```

```
/' lista bat memoria
dinamikoaz osatzea
#include<stdlib.h>
main ( )
{  int n,i,zenb;
   int `lista
   printf ("zenbat balio?:");
   scanf ("%d",&n);
   list = (int') malloc (n*sizeof  (int));

       scanf ("%d, list +i);
```

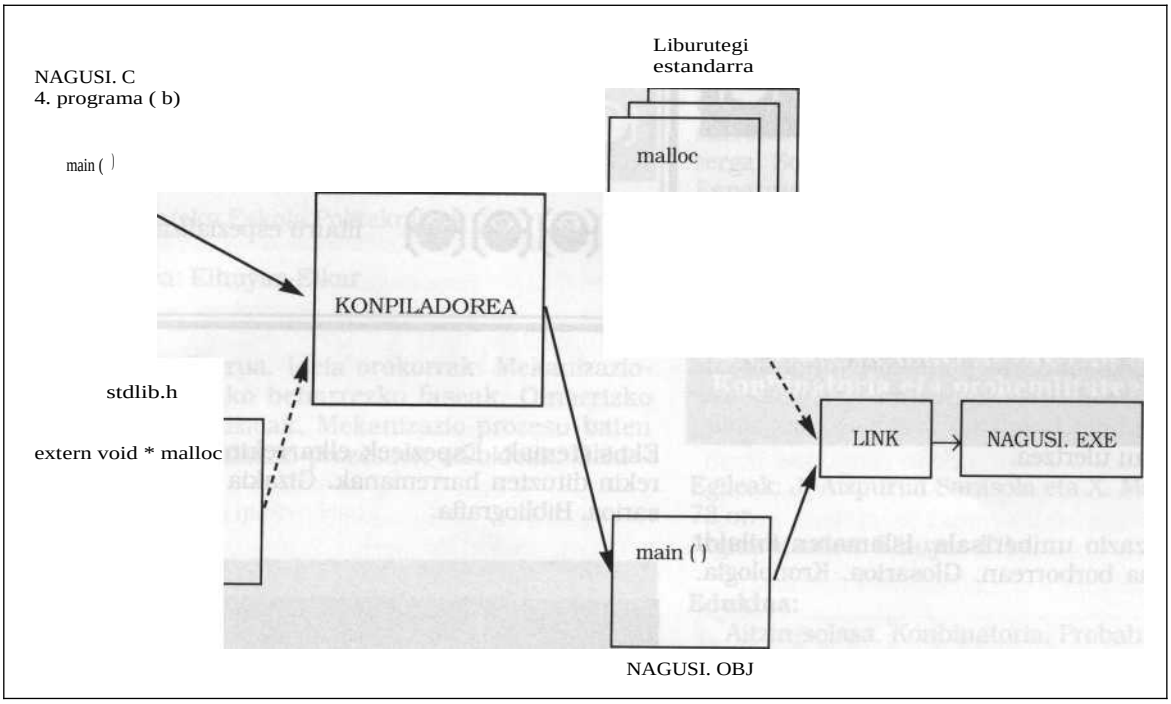
4. programa , Memoria dinamikoaren erabilpena.

batean kataloga daitezke, estekatzaileak erabil ditzan erreferentziatzen dituzten programekin. Horretarako 9. kapituluan aztertutako aldagaien ezaugarriak kontutan hartu behar dira.

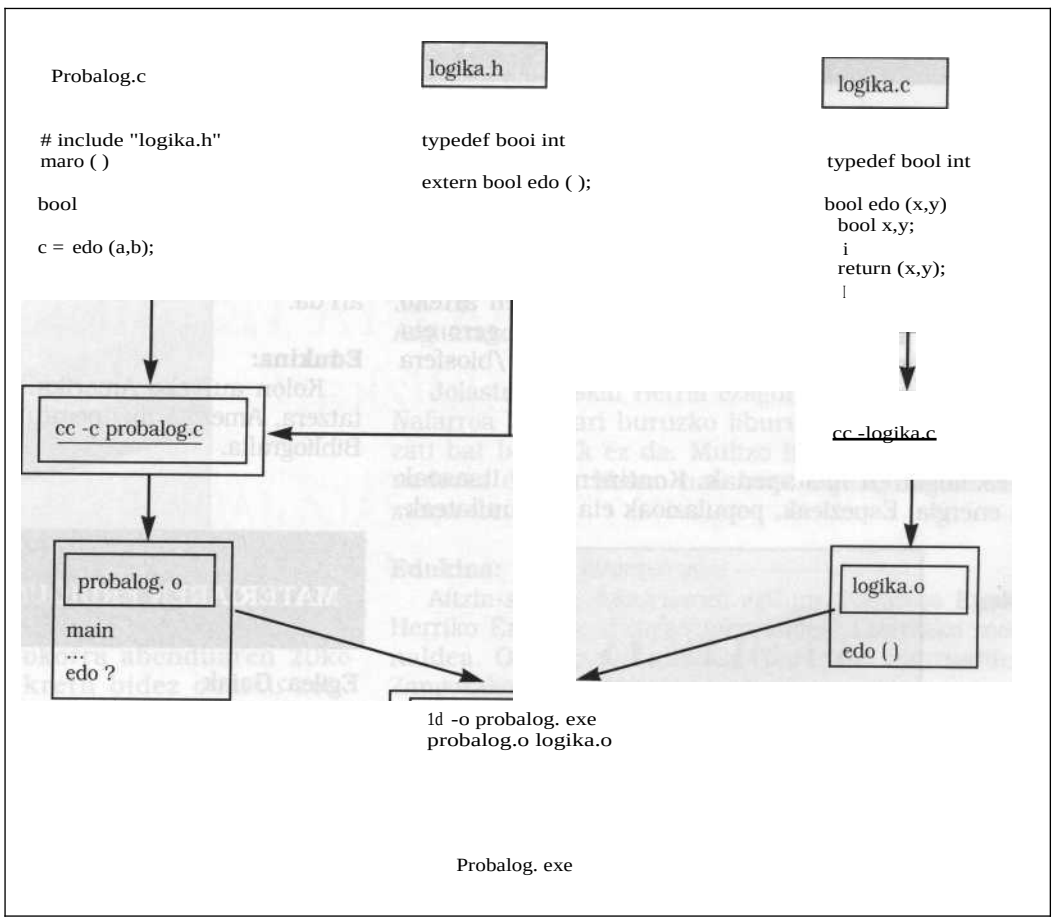
Bide honetan oso komenigarria da objektu-modulu edo liburutegi bakoitzeko izenburu-fitxategi bat egitea, bertan konstante eta datu-moten definizioa zein funtzioen erazagupena egingo delarik. 2. eskeman,

1. taula. Funtzio estandar garrantzitsuak.

Funtzio-prototipoa	Izenburu-fitxategia	Esanahia
Karaktere-motei buruz		
int tolower (int)	ctype. h	minuskula bihurtu
int toupper (int)	ctype. h	maiuskula bihurtu
int islower (int)	ctype. h	baldin minuskula egiazkoa
int isupper (int)	ctype. h	baldin maiuskula egiazkoa
int isalpha (int)	ctype. h	baldin letra egiazkoa
int isdigit (int)	ctype. h	baldin digitu egiazkoa
Funtzio matematikoak		
int abs (int)	stdlib.h	balio absolutua (osoa)
double pow (double, double)	math . h	berreketa
double sqrt (double)	math . h	erro karratua
double fabs (double)	math . h	balio absolutua (erreala)
double logia (double)	math . h	logaritmo hamartarra
double sin (double)	math . h	sinua
double cos (double)	math . h	cosinua
double tan (double)	math . h	tangentea
double asin (double)	math . h	arku sinua
double acos (double)	math . h	arku cosinua
double atan (double)	math . h	arku tangentea
Memoria dinamikoari buruz		
void ` malloc (int)	stdlib . h	parametroak zehaztutakoa adina karaktere eskuratu
void * free (void ")	stdlib . h	parametroak zehazten duen memoria-zatia askatu
void " calloc (int)	stdlib . h	malloc bezala, baina bi parametro batek elementu--kopurua eta besteak elementuaren luzera adieraziz
Karaktere-kateei buruz		
char * strcpy (char*,char*)	string . h	katea osoa kopiatu
char * strncpy (char*,char* int)	string . h	katea kopiatu luzera batez
char' strcat (char*,char*)	string . h	2 katea bildu
int strcmp (char*,char*)	string . h baldin <-1	baldin >1, baldin=0 eta
char * strchr (char*,char)	string . h	karaktere bat agertzearen posizioa
Bestelakoak		
int atoi (char *)	stdlib . h	karaktere-katea zenbaki oso bihurtu
float atof (char*)	stdlib . h	karaktere-katea zenbaki erreal bihurtu
int rand (void)	stdlib . h	ausazko zenbakia sortu
void bsearch (...)	stdlib . h	osagai bat bilatzea
void qasort (...)	stdlib . h	taula baten sailkapena



1. eskema. Liburutegi estandarraren erabilpena.



2. eskema. Konpilazio banatua.

ikusitako 2. programari dagokion konpilazio-estekaketaren eskema azaltzen da.

Bigarren eskema horretan **logika.o aparteko** modulua da, baina ez da liburutegi batean katalogatu. Horretarako ar (UNIXekin) programa erabili behar da eta estekatzeko komandoa aldatu. Geldituko litzatekeen komando-multzoa honako hau litzateke:

```
cc - c logika .c
ar rc lib logika.a logika.o
cc -o probalog.exe probalog.c -l logika
```

E