7. GAIA

KARAKTERE KATEAK - STRING

1 SARRERA

Karaktere kateak edo string-ak taula numerikoak edo array-ak bezalako datu egiturak dira, baina ezberdintasun batekin: zenbakiak gorde beharrean karaktereak gordetzen dituzte.
Nahiz eta orain arte arrayentzat ikusitako guztia stringentzat ere balio izan, karaktere kateek taula numerikoei aplikagarriak ez diren berezitasun propioak aurkezten dituzte, eta hurrengo atalean azalduko dira. Orokorrean, strringak bi modutara maneiatu daitezke:

- Taula numerikoak bezala.
- Stringen berezitasun propioetara joz.

Kasu batzutan, lehenengo aukera hautatzea beste erremediorik ez da egongo, baina gehienetan karaktere kateen berezitasun propioak erabiliko dira planteatzen den arazoa ahalik eta modu errazenean ebazteko.
String baten deklarazioa eta array batena oso antzekoak dira, nahikoa delako char tipoko elementuak dituen bektore bat definitzearekin, izen bat eta elemnetu kopurua ere adieraziz.

```
char string_izena [elemento_kopurua];
```

Adibidea

Demagun gehienez 30 elementu dituen string bat erabili nahi dela, katea1 izenarekin. Honela deklaratu beharko litzateke:

```
char katea1[30];
```

Ondorioz, memorian char tipoko 30 elementurentzat erresebatuko da legua, honela adierazita:

katea1

¿?	¿?	¿?	¿?	¿?	¿?	¿?	¿?	¿?	¿?		¿?	¿?	¿?	¿?
0	1	2	3	4	5	6	7	8	9	…		27	28	29

Taula numerikoekin gertatzen den bezala, karaktere kate bat deklaratzean, posizio bakoitzan esleiten den hasierako balioa ezezaguna da (zaborra). Behin stringa deklaratuta dagüela, bere posizio bakoitza char tipoko aldagai bat balitz bezala erabili daiteke.

2 STRINGEN BEREZITASUN PROPIOAK

Sarreran aurreratu den moduan, karaktere kateek taula numerikoak ez bezala erabiltzen ahalbidetzen duten berezitasun propioak dituzte. Argi dago arrayek berezitasun hoiek aurkezten ez dituztela, beraz, ez dauka zentzurik karaktereena ez den bektore batin erabiltzen saiatzea.

2.1 Kate amaieraren karakterea ('\0')

Orain arte char tipoko elemento batek hartu ditzakeen balio batzuk ikusi ditugu; letraz ('a'...'z' y 'A'...'Z) eta digitoez ('0'...'9') gain, karaktere ez alfanumerioen saila bat dago, eta horien artean hurrengoak aipatu daitezke: puntuaziokoak ('.', ',', eta abar), banatzaileak (' ', '\n' y '\t') eta teklatuko bereziak ('*', '|', '@', y '€', batzuk aipatzeagatik).
Kate amaierako karakterea bezala ezagututako balio bat ere badago, '\0' legez adierazten dena eta oso garrantzitsua dena stringak erabiltzean. Kate amaierako karaktereak karaktere katearen azkenengo posizioa adierazten du.

katea1

'H'	'o'	'l'	'a'	'\0'	¿?	¿?	¿?	¿?	¿?		¿?	¿?	¿?	¿?
0	1	2	3	4	5	6	7	8	9	…		27	28	29

Honela, string baten azkenengo balio egokiaren atzetik beti '\0' bat jartzen bada, ez dugu jakin behar izango zenbat elementu dauden katean (arrayetan gertatzen zen bezala); katearen posizio guztiak aztertu besterik ez da egin behar izango kate amaaireako karakterea aurkitu arte, momentu horretan azkenengo balio onargarria ailegatu izango da.
String bateko karaktereak banan banan gordetzen direnenan, '\0' karakteea azkenengo posizioan sartzearen ardura programadorearena da.

Adibidea

```
char katea1[30];

katea1[0] = 'H';
katea1[1] = 'o';
katea1[2] = 'l';
katea1[3] = 'a';
katea1[4] = ' ';   /* zuriunea */
katea1[5] = 'm';
katea1[6] = 'u';
katea1[7] = 'n';
katea1[8] = 'd';
katea1[9] = 'o';
katea1[10] = '\0'; /* kate amaierako karakterea */
```

Aginduak bete ondoren, katea1 stringa honela geratzen da:

katea1

'H'	'o'	'l'	'a'	' '	'm'	'u'	'n'	'd'	'o'	'\0'	¿?		¿?	¿?
0	1	2	3	4	5	6	7	8	9	10	11	…	28	29

Kasu askotan programadoreak ez du '\0' karakterea esplizitoki sartu behar, zeren eta, georoxeago ikusiko den moduan, egoera hauetan kate amaiera autmatikoki ezartzen da daokion posizioan.

2.2 Karaktere kate baten hasieraketa

Arrayekin gertzatzen den bezala, karaktere kate baten lehenengo posizioek hasierako balioak hartu ditzakete stringaren deklarazioaren momentuan. Adibidez:

```
char kat1[20] = "Urano";
```

kat1

'U'	'r'	'a'	'n'	'o'	'\0'	¿?		¿?	¿?
0	1	2	3	4	5	6	…	18	19

Gogoratu esleipen mota hau bakarrik karaktere katearen deklarazioan egin daitekela, eta beste momentu batetan egiten bada, konpilaketa errore bat emango du.
Ikus daiteke, aurrekoaren ondorioz, kat1 katearentzat memorian lekua erreserbatu dela eta bere lehenengo posizioak hasieratuak izan direla: lehenengo bostak "Urano" hitza osatzen dituen karaktereekin eta seigarrena kate amaierako karakterearekin, azken hau automatikoki gehitzen delarik.

2.3 Stringen irakurketa teklatutik: scanf eta gets funtzioak

Aurreko gaian ikusi den bezala, taula numeriko bat erabiltzaileak tekleatzen dituen balioekin betetzeko, beharrezkoa da datuak banan banan harrapatzea, scanf agindua bukle batetan sartuz. Irakurri nahi den elementu kopurua aurretik jakinik ezean, normalena kontadore bat definitzea da, arrayan sartu den balio kopurua zenbatzeko eta definitutako gehienezkoa baino elementu gehiago sartzea eragozteko. Kontadore honek ere geroagoko arrayerako atzipena controlatzeko balio du.
Stringen kasuan, arrayena bezalako irakurpen bat proposatu daiteke, hau da, karakterez karaktere. Adibidez, horrela:

```
int i=0;
char kar, katea[30];
do
{
   printf("Sartu karaktere bat (puntua('.') amaitzeko)");
   fflush(stdin);          /*teklatuaren buffera garbitzen du */
   scanf("%c", &kar);
   if (kar != '.')
   {
     katea[i] = kar;
     i= i+1;
   }
}while (i<30 && kar != '.');
katea[i]= '\0';     /*kate amaierako karakterea gehitu */
```

Nahiz eta stringak gordetzen dituen egiturak arrayak bezala funtzionatu, ohizkoena karaktere kateen berezitasunak erabiltzea da. Izan ere, stringak unitate bat bezala hartu daitezke, eta batean osorik irakurtzea ahalbidezen duten sarrera funtzio zehatzak daude, aurreko ebazpenak aurkezten dituen arazo nagusiak sahiestuz, hots: kate bat batean irakrri daiteke eta ez karakterez karaktere, karaktere berri bakoitza irakurri baino lehen teklatuaren buffera garbitu behar izatea ekiditzen da, eta ez da kate amaierako karakterea eskuz gehitu behar..

Aurretik aurkeztu eta erabilitako scanf funtzioa kate baten gordeko diren karaktereak batean irakurtzeko erabili daiteke. Horretarako, "%s" kodearengana joko dugu, atzeman nahi dena string bat dela adierazteko.

```
char katea[20];
...
scanf("%s", katea);
```

Behatu hau dela kasu bakarra non scanf funtzioaren parametroaren aurretik ampersand (&) sinboloa ez doana.
Aurreko aginduak erabiltzaileak tekleatzen dituen karaktere guztiak irakurri eta *katea* stringaren posiziotan gehituko ditu, 19 elementura heldu arte (kasu horretan, soberan daudenak teklatuaren bufferean geratuko dira, irakurrita izatearen zain) edo zuriune bat (' '), tabulazio bat ('\t') edo linea berri bat ('\n', intro edo return) atzeman arte.

Gets funtzioa eragiketa bera betetzen du, baina ezberdintasun batekin: zuriune edo tabulazio bat irakurtzean ez da hor baratzen, baizik eta keteari atxikitzen dio beste karaktere bat balitz legez. Horregatik hain zuzen ere, ohizkoagoa da kate irakurketak gets funtzioarekin egitea scanfrekin baino.
Kaaktere kate bat irakurtzean, bi funtzioak, scanf eta gets, kate amaierako karakterea dagokion posizioan sartzen dute.

Adibidea:

Jarraian, C kode adibide bat erakusten da, erabiltzaileari karaktere sekuentzia bat eskatzen diona eta sting batetan gordetzen duena.

```
char karkat[30];

printf("Sartu gehienez 29-ko karaktere sorta:\n");
fflush(stdin);
gets(karkat);
```

Exekuzio adibidea:

Hurrengo irudiak kode zati txiki hau exekutatzean pantailan idatziko litzatekeen mezua erakusten du. Karaktere kate bat eskatzean, erabiltzaileak nahi duena tekleatu ahal izan beharko du, return edo intro teklarekin bukatuz, esan den moduan gets aginduak karaktereak atzitzeari uzteko tekla bakarra delako. Demagun erabiltzaileak tekleatzen duen testua, eta tipografia kursiva eta azpimarratuan agertzen dena, "Mercurio" hitza dela.

Pantaila

```
Sartu gehienezko 29 karaktereko katea:
Mercurio
```

Karkat katea honela geratuko litzateke

karkat

'M'	'e'	'r'	'c'	'u'	'r'	'i'	'o'	'\0'	'¿?'		¿?	¿?
0	1	2	3	4	5	6	7	8	9	…	28	29

Adibidean ikusten den bezala, programadoreak ez du '\0' karaktereaz arduratu behar, gets agindua betetzean, ordenagailuak zuzenean dagokion posizioan sartzen duelako.

2.4 Stringen idazketa pantailan. Printf funtzioa

Taula numeriko baten elementuak pantailan idazteko, beharrezkoa da arrayaren posizio baliogarri guztiak aztertzea, elementu guztiak banan banan erakusteko, aurreko adibidetan ikusi den moduan. Azken batean, iterazio bakoitzan pantailan balio bakar bat imprimatzen duen bukle bat implementatzea beharrezkoa da.
Nahiz eta karaktere kateen kasuan ere stringaren elementuak horrela idatzi ahal izan, kate amaierako karaktereari esker, katearen elementu baliogarri guztien idazketa egitea posible da printf agindu bakar batekin, hau da, bukle bat implementatu barik.
Aurreko atalean scanf aginduari buruz komentatu denatik aurrera, ez da arraroa hurrengo aginduak autamatikoki eta batean pantailan erakustea '\0' karakterearen aurretik (eta hau ez da idazten) dauden karkat katearen balioak.

```
printf("%s", karkat);
```

Exekuzio adibidea:

Jarraian erabiltzaileari izena eskatzen dion kodea erakusten da, geroago pertsonalizatutako agurra emateko. Lehen bezala, erabiltzileak tekleatzen duen testua letra kursibo eta azpimarratuan dator, programak pantailan erakusten dituen mezuengandik bereizteko..

```
char izena[30];

printf("Sartu zure izena, mesedez: ");
fflush(stdin);
gets(izena);
printf("Kaixo, %s.", izena);
```

Pantaila

```
Sartu zure izena, mesedez: Maite
Kaixo, Maite.
```

Izena katea, honela geratuko litzateke:

izena

'A'	'n'	'a'	' '	'R'	'o'	's'	'a'	'\0'	'¿?'		¿?	¿?
0	1	2	3	4	5	6	7	8	9	…	28	29

2.5 String bateko elementu baliogarrien kopurua: strlen funtzioa

Karaktere kate baten azkenengo posizio baliogarria '\0' karaktereak betetzen duenez, programadoreak itxaroten den gehienezkoa baino posizio bat gehigorekin deklaratu beharko ditu stringak. Horrela, gehienez 9 karaktere betetzen dituen NAN bat gorde nahi bada, dagokion stringak 10 karaktere izan beharko ditu (8 digito, letra bat eta '\0' karakterea).

Adibidea:

```
char nan[10];

printf("Sartu zure NAN-a, mesedez: ");
fflush(stdin);
gets(nan);
```

Pantaila

```
Sartu zure NAN-a, mesedez: 14278210P
```

Nan katea kasu honetan, beteta geratuko litzateke bere azkenengoposizio fisikorarte:

nan

'1	'4'	'2'	'7'	'8'	'2'	'1'	'0'	'P'	'\0'
0	1	2	3	4	5	6	7	8	9

Adibide honetan erabiltzaileak zehazki 9 karaktereak tekleatu dituela ikusi da, eta hori da stringak gorde ahal dituen gehienezko karaktereak, '\0' karakterea konttan hartu gabe. 9 karaktere baino gutxiago tekleatu izan balira, kate amaierako karakterea beste ezkarragoko posizio batetan kokatuko litzateke, *Mercurio* eta *Maite* adibidetan legez. Arazoak datoz erabiltzaileak 9 karkatere baino gehiago tekleatzen dituenean, lehenengo 9 bakarrik atzematen direlako, azkenengo posizioan '\0' karaktereak gordez, soberako karaktereak teklatuaren bufferean geratzen direlarik, irakurrita izateko zain (edo fflush aginduarekin ezabatuak izateko zain).

String baten irakurketa karakterez karaktere egiten denean oso erraza da jakitea zenbat karaktere irakurri diren teklatutik. Egoera honetan, karaktere bakoitza non kokatuko den adierazten duen aldagai bat erabiltzea beharrezkoa denez (2.3 atalean i deitzen dena), behin elementuen atzipena amaituta, aldagai bera izango da esango diguna zenbat karaktere atzitu diren.
Hala ere, string baten irakurketa zuzenera jotzen denean, hots, agindu bakar baten bidez (scanf edo, orokorrean, gets) teklatutik sartutako karaktere guztien atzematera, katean gordetako karaktere kopurua ezezaguna. Eskerrak, irakurketa aginduak kate amaierako karaktera gehituko duelari erabiltzaileak tekleatutako azkenengo balioaren ostean.Kontutan hartuz arrayaren posizioak 0tik aurrera zenbakitzen direla, '\0' karakterea dagoen posizioa irakurri diren elementuen kopurua adieraziko du. Honela, aurreko adibidea berrikusten bada *Maite* katea batean atzematen zenean, ikus daiteke nola kate amaiera bostgarren posizioan zegoela, eta hain zuzen ere, 5 ziren irakurritako karaktere baliagarriak.

izena

'A'	'n'	'a'	' '	'R'	'o'	's'	'a'	'\0'	'¿?'		¿?	¿?
0	1	2	3	4	5	6	7	8	9	…	28	29

String baten karaktere baliogarrien kopurua identifikatzeko ataza errazteko, strlen deritzon funtzioa dago, eta karaktere kate bat hartzen du sarrera parametro bezala eta balio oso bat itzultzen du '\0' karakterearen posizioa adierazten duena, hau da, sarrerako karaktere katearen karaktere baliogarrien kopurua. Apur bat aurrerago ikusiko da adibide moduan strlen funtzioarren inplementazio posible bat.

Adibidea:

Jarraian erabiltzailearen izena eskatzen duen programaren kodea luzatzen da, geroago agur pertsonalizatua idazteko, bere izena zenbat letra dituen esaten duen mezu txiki baten bitartez. Lehen bezala, erabiltzaileak tekleatzen duen testua letra kursibo eta azpimarratuan doa, programak pantailan idazten dituen mezuengandik bereizteko.

```
char izena[30];
int luzera;

printf("Sartu zure izena, mesedez: ");
fflush(stdin);
gets(izena);
luzera = strlen(izena);
printf("Kaixo, %s. Zure izenak %d karaktereko luzera du.", izena,
luzera);
```

Pantaila

```
Sartu zure izena, mesedez: Maite
Kaixo, Maite. Zure izenak 5 karaktereko luzera du.
```

3 KARAKTERE KATEAK ETA FUNTZIOAK

String tipoko argumentua hartzen duen funtzio bat garatzeko momentuan, taula edo bektore numerikoei buruz esandako guztia hartu behar da kontutan.

- Funtzioaren erabileran, parametro errealera joteko, inoiz ez dira kortxeteak erabiltzen. Hartu adibide bezala aurreko atalean erabilitako gets eta strlen funtzioak.
- Funtzioaren prototipo eta definizioan parametro formala karaktere kate bat dela adierazi behar da, nahiz eta bere luzera adierazi ez behar. Horretarako nahikoa izango da [] erabiltzearekin, hau da, kortxeteak hutsik.

Adibidea: *strlen* funtzioa

Jarraian strlen funtzioaren prototipo eta definizio posible bat aurkezten da. Aurretik ikusi den moduan, karaktere kate bat hartzen du sarrera bezala eta zenbaki oso bat itzultzen du. Zenbaki hau '\0' karakterea string-ean okupatzen duen posizioa da, hau da, sarrerako kateak duen karaktere erabilgarrien kopurua.

Strlen funtzioaren prototipoa hau izan leike:

```
int strlen (char kat[]);
```

Oraintxe gogoratu denez, string-aren elementu kopurua ez da zehaztu behar, ezta prototipoa ezta funtzioaren definizioa ere. Hala ere, karaktere katearentzat tamaina bat adierazi izan leike, kasu honea "cad[]" erabili da, abantaila handia eskeintzen duena: katea definitzeko erabili den tamaina edozein dela (katea[30], nif[10], izena[N],...) strlen funtzioa bere tamaina erabilgarria kalkulatzeko balio izango du.

Una posible definición o cuerpo de la función *strlen* es la que se presenta a continuación:

```
int strlen (char kat[])
 {
   int i;
   i = 0;
   while (kat[i] != '\0')
    {
      i = i + 1;
    }
   return (i);
 }
```

Ikus daiteke nola, funtzioan bilatu eta itzuli behar den balioa '\0' karakterea dagoen posizioarekin bat datorren moduan, egin behar den bakarra i indize bat hasieratzea eta handitzea da taulak, i-n indexatuta, kate amaierako karakterea daukan lehenengo balioa aurkitu arte. Argi dagoenez, posizio hori existitzen ez bada, hau da, stringeko posizio bakar bat ere ez badauka '\0' karakterea, orduan hemen aurkeztutako inplementazioa arazoak emango lituzke. Hala ere, lehenengo momentutik suposatu da egoera hori ezin dela eman, bai katea gets edo scanf erabiliz hasieratu delako, bai erabilera programatu duena kate amaiera esplizitoki sartzeaz ziurtatu delako

3.1 string.h libreria

Karaktere kateekin lan egiteko orduan hurrengo lerroa gehitzea beharrezkoa izango da:

```
#include <string.h>
```

Lerro hau programa batetan sartzea stringen tratamendurako definitutako funtzioak erabiltzea ahalbidetzen du, eta haien artean hauek aipagarriak dira:

int strlen(char kat[]);

- Iadanik ikus denez, funtzio honek karaktere kate baten luzera erabilgarria itzultzen du, eta hau '\0' karakterearen posizioarekin bat dator.

void strcmp (char kat1[ ], char kat 2[ ]);

- Funtzio honek kat1 eta kat2 kateak konparatzen ditu eta 0 balioa itzultzen du berdinak badira, Ezberdinak badira, beste balio bat itzultzen du, sarrerako string bien edukiaren arabera positiboa ala negatiboa izanik.

void strncpy (char kat 1[ ], char kat 2[ ], int n);

- Funtzio honek kat2-ren lehenengo n karaktereak kat1-en kopiatzen ditu.

4 EBATZITAKO ARIKETA

4.1 Bi karaktere kate berdinak diren esaten duen programa

Ariketa hau erabiltzaileari bi karaktere kate eskatzean datza, bakoitza string banan gordeko delarik, eta ondoren biak berdinak diren ezarriko da. Kateek 50 karaktere baino gehiago ez dituela suposatuko dugu.
Ideia nagusia zera da, behin irakurrita, kateek karaktere kopuru berdina ez badute, ezberdinak direla esan nahiko du, eta kontrako kasuan (hau da biak luzera berakoak dira) bakarrik hartuko dira berdintzat zehazki karaktere berdinak ordena berdinean badaude.
Ariketa interesgarriagoa egiteko, ez da strcmp funtzioa erabiliko. Horren ordez, hurrengo prototipoa duen funtzioa idatziko da:

int berdinak_dira(char kat1[ ], char kat2[ ]);

- Funtzio honek kat1 eta kat2 kateak konparatu, eta 1 balioa itzultzen du berdinak badira edo 0 balioa ezberdinak badira. Suposatzen da sarrerako kate biak (kat1 eta kat2) kate amaierako karakterea daukatela.

Ebazpen posible bat jarraian aurkezten da:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int berdinak_dira(char kat1[], char kat2[])
 {
   int luz1, luz2, i, berdinak;

   luz1 = strlen (kat1);
   luz2 = strlen (kat2);

   if (luz1 != luz2)
    {
      return(0);
    }
   else
    {
      i = 0;
      berdinak = 1;
      while ((berdinak == 1) && (i<luz1))
       {
         if (kat1[i] != kat2[i])
          {
            berdinak = 0;
          }
         i = i + 1;
       }
      return (berdinak);
    }
 }
```

```
void main()
{
  /* aldagaien deklarazioa */
  char karkat1[51];
  char karkat2[51];
  int erabakia;
```

```
  /* sarrerako kateen eskaria */
  printf ("\nIdatzi lehenengo katea (gehienez 50: ");
  gets (karkat1);
  printf ("\nIdatzi bigarren katea (gehienez 50: ");
  gets (karkat2);

  /* berdinak dira? */
  erabakia = berdinak_dira(karkat1, karkat2);

  /* Emaitzaren aurkezpena */
  if (erabakia == 1)
   {
     printf("\n%s y %s kateak berdinak dira.",karkat1,karkat2);
   }
  else
   {
     printf("\n%s y %s kateak ezberdinak dira.",karkat1,karkat2);
   }
  system("pause");
}
```

Adibide honetan, programa nagusiak orainarte ikusitako ohizko pausuak burutzen ditu: erabiltzaileari balioa eskatzen dizkio, *berdinak_dira* funtzioa erabiltzen du erantzun bat lortzeko eta emaitzaren arabera, mezu bat edo bestea inprimatzen du pantailan. *berdinak_dira* funtzioa, bere aldetik, lehenengo parametro moduan heldutako kateen luzerak ebaluatzen ditu. Ezberdinak badira, 0 bihurtzen du, kateak berdinak ez direla adierazteko. Kontrako kasuan, *berdinak* aldagaia 1 balioarekin hasieratu eta posizioz posizio kate bien karaktereak konparatzen ditu. Ezberdinak diren bi posizio aurkitzen ba ditu, *berdinak* 0 balioa hartzen du. Horrela, behin kate biak zeharkatu direla, *berdinak* 1 balioa badauka diferentziarik ez dela aurkitu esan nahi du.