PROGRAMAZIOA
C programazio lengoaia 3

1) Erakusleak
2) Erabilpen kasuak
3) Adibideak

1.1 ERAKUSLEAK

- Memoria-helbideak dira
- Helbide horien bitartez memoria atzitu ahal da. Aldagaien balioak erabili edota aldatu ahal dira.
- Egokiak dira:
  - Azpiprogramek **emaitza anitz bueltatu** ditzaten
  - Azpiprogramei **taulak pasatzeko**
  - **Taulak korritzeko**

1.2 ERAKUSLEAK
Sortu eta erabili

- Erazagupena:
  mota *izena; **float *erakF = &Soldata;**
- Erakuslearen helbidea aldatzea:
  izena = helbidea; **erakF = erakF - 5 ;**
- Erakusleak apuntatzen duen memoria guneko balioa aldatu:
  ***izena = balioa;** ***erakF = 125.68;**
- Erakusleak apuntatzen duen memoria guneko balioa erabili:
  … = … *izena …; **B = *erakF * 2;**

1.3 ERAKUSLEAK
Erakusleei erakusleak

- Erazagupena:
  mota *izena; **mota **izena;** **mota ***izena;**
  float *erakF = &Soldata;
  float **erakErakF = &erakF;
  float ***erakErakErakF = &erakErakF;
- Erakuslearen helbidea aldatzea:
  izena = helbidea;
  erakErakF = *erakErakErakF;
- Erakusleak apuntatzen duen memoria guneko balioa aldatu:
  ***izena = balioa;** **izena = balioa;** ***izena = balioa;**
  erakErakF = 125.68;
  ***erakErakErakF = 200.22;**

1.4 ERAKUSLEAK
Erakusle taulak

- Erazagupena:
  mota *izena[dimentsio1];
  int A=0, B= 1, C= 2, D= 3;
  int *Terak[3] = {&A, &B, &C};
- Erakuslearen helbidea aldatzea:
  izena = helbidea;
  Terak[0] = &D
  Terak[1]= Terak[2];
- Erakusleak apuntatzen duen memoria guneko balioa aldatu:
  ***izena = balioa;**
  ***Terak [1] = 20;**

1.5 ERAKUSLEAK
Kontuz!!!

- Kontuz izartxoekin:
  - Biderkaketa:
    A = Kont * 5;
  - Erakusle erazupena:
    int *erak, **erakerak;
  - Erakuslearen helbide atzitu:
    ***erak = 5;**
  - Erakusleak apuntatzen duen erakuslearen helbidea atzitu:
    Soldata = **erakerak;
  - Dena nahastuta:
    Soldata = *erak ***erakerak;

2.1 ERABILPEN KASUAK
1. Kasua: Aldagai atzipena

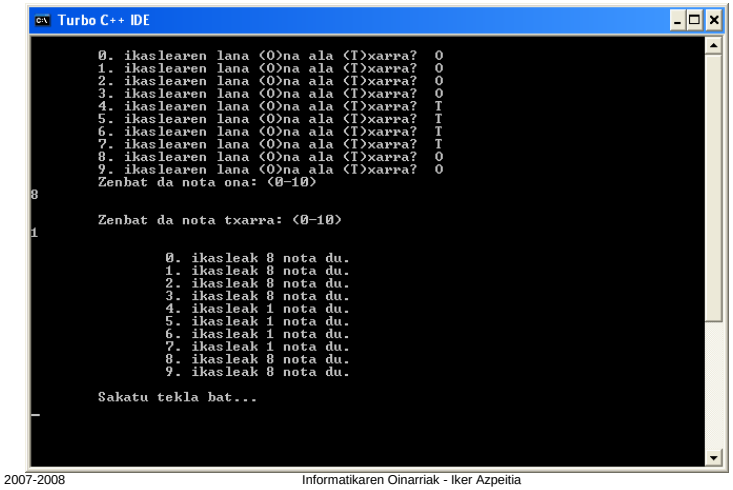
```
void main ()
{float Soldata1, Soldata2;
float *erakF = &Soldata1;
*erakF = 2000.50;
erakF = &Soldata2;
*erakF = Soldata1 + 444;
}
```

…	?	
65000	2000.50	Soldata1
65004	2444.50	Soldata2
65008	65004	erakF
65012		
…		

2.2 ERABILPEN KASUAK
2. Kasua: erakusle taula

```
#include <conio.h>
#include <stdio.h>

void PantailaratuNotak (int * *erak, int topea);

void main()
{char Erantzuna;
int  i, NotaOna, NotaTxarra;
int *NotaTaula [10];

clrscr();

for (i = 0; i < 10; i++)
     {do{
          printf ("\n\t d. ikaslearen lana (O)na ala (T)xarra? ", i);
          Erantzuna = getche();
     }while (Erantzuna != 'O' && Erantzuna != 'T');

     if (Erantzuna == 'O')
               {NotaTaula[i] = &NotaOna;}
     else
               {NotaTaula[i] = &NotaTxarra;}
     }
do{
     puts ("\n\t Zenbat da nota ona: (0-10)" );
     scanf ("%d", &NotaOna);
}while (NotaOna < 0 || NotaOna >10);

do{
     puts ("\n\t Zenbat da nota txarra: (0-10)" );
     scanf ("%d", &NotaTxarra);
}while (NotaTxarra < 0 || NotaTxarra >10);

PantailaratuNotak(&NotaTaula[0], 10);

puts ("\n\n\tSakatu tekla bat...");
getch();
}

void PantailaratuNotak (int * *erak, int topea)
{ int k;
for (k=0; k <topea; k++)
  {printf ("\n\t%d. ikasleak %d nota du.\n", k, *(*(erak+k)));
  }
}
```

2.3 ERABILPEN KASUAK
2. Kasua: erakusle taula



2.4 ERABILPEN KASUAK
3. kasua: erakusleei erakusleak

```
PantailaratuNotak(&NotaTaula[0], 10);

void PantailaratuNotak (int * *erak, int topea)
{ int k;
for (k=0; k <topea; k++)
    {printf ("%d. ikasleak %d nota du.\n", k, *(*(erak+k)));
    }
}
```

65000	?	Erantzuna
65001	?	i
65003	8	NotaOna
65005	1	NotaTxarra
65007	65003	NotaTaula[0]
65009	65003	NotaTaula[1]
65011	65003	NotaTaula[2]
65013	65003	NotaTaula[3]
65015	65005	NotaTaula[4]
65017	65005	NotaTaula[5]
65019	65005	NotaTaula[6]
65021	65005	NotaTaula[7]
65023	65003	NotaTaula[8]
65025	65003	NotaTaula[9]
65027	65007	erak
65029	10	topea

2.5 ERABILPEN KASUAK
4. kasua: Taulak korritu

int T[4][3]={{0,1,2},{10,11,12},…};

int *erak = &T[0][0];

Lerroa\Zutabea	0	1	2
0	0	1	2
1	10	11	12
2	20	21	22
3	30	31	32

…	?	
65000	0	T[0][0]
65002	1	T[0][1]
65004	2	T[0][2]
65006	10	T[1][0]
65008	11	T[1][1]
65010	12	T[1][2]
65012	20	T[2][0]
65014	21	T[2][1]
65016	22	T[2][2]
65018	30	T[3][0]
65020	31	T[3][1]
65022	32	T[3][2]
65024	65000	erak
…	?	

2.6 ERABILPEN KASUAK
4 kasua: Taulak korritu

- Taula osoa korritu

```
for (erak=&T[0][0]; erak < = T[3][2]; erak++)
   {printf ("%d", *erak);}
```

- Bigarren lerroa

```
for (erak=&T[0][0] + 2 * 3 , i=0; i < 3; i++)
   {printf ("%d", *(erak+i));}
```

- Bigarren zutabea

```
for (erak=&T[0][0] + 2 , i=0; i < 4; i++)
   {printf ("%d", *(erak+ i * 3));}
```

- Lau puntak

```
erak=&T[0][0];
printf ("%d", *erak);
printf ("%d", *(erak+(3-1)));
printf ("%d", *(erak+(4-1)*3));
printf ("%d", *(erak+(4*3)-1));
```

…	?	
65000	0	T[0][0]
65002	1	T[0][1]
65004	2	T[0][2]
65006	10	T[1][0]
65008	11	T[1][1]
65010	12	T[1][2]
65012	20	T[2][0]
65014	21	T[2][1]
65016	22	T[2][2]
65018	30	T[3][0]
65020	31	T[3][1]
65022	32	T[3][2]
65024	…	erak
…	?	

2.7 ERABILPEN KASUAK
5 kasua: m*n taula orokorra

char KarT[m][n];

Lerroa\Zutabea	0	…	n-1
0	**0**		**n-1**
…			
m-1	**(m-1)*n**		**(m*n)-1**

…		…
65000		KarT[0][0]
…		…
65000+n-1		KarT[0][n-1]
65000+n		KarT[1][0]
…		…
…		…
65000+(m-1) * n		KarT[m-1][0]
…		…
65000+(m*n)-1		KarT[m-1][n-1]
…		

Taulan gordeta dauden balioak, KarT[0][0] elementutik zenbat salto egin behar diren adierazten dute.

2.8 ERABILPEN KASUAK
5 kasua: m*n taula orokorra

- Taula osoa korritu

```
for (erak=&T[0][0]; erak < = T[m-1][n-1]; erak++)
   {printf (“%d”, *erak);}
```

- J lerroa

```
for (erak=&T[0][0] + J * n , i=0; i < n; i++)
   {printf (“%d”, *(erak+i));}
```

- K zutabea

```
for (erak=&T[0][0] + K , i=0; i < m; i++)
   {printf (“%d”, *(erak+ i * n));}
```

- Lau puntak

```
erak=&T[0][0];
printf (“%d”, *erak);
printf (“%d”, *(erak+n-1));
printf (“%d”, *(erak+(m-1)*n));
printf (“%d”, *(erak+(m*n)-1));
```

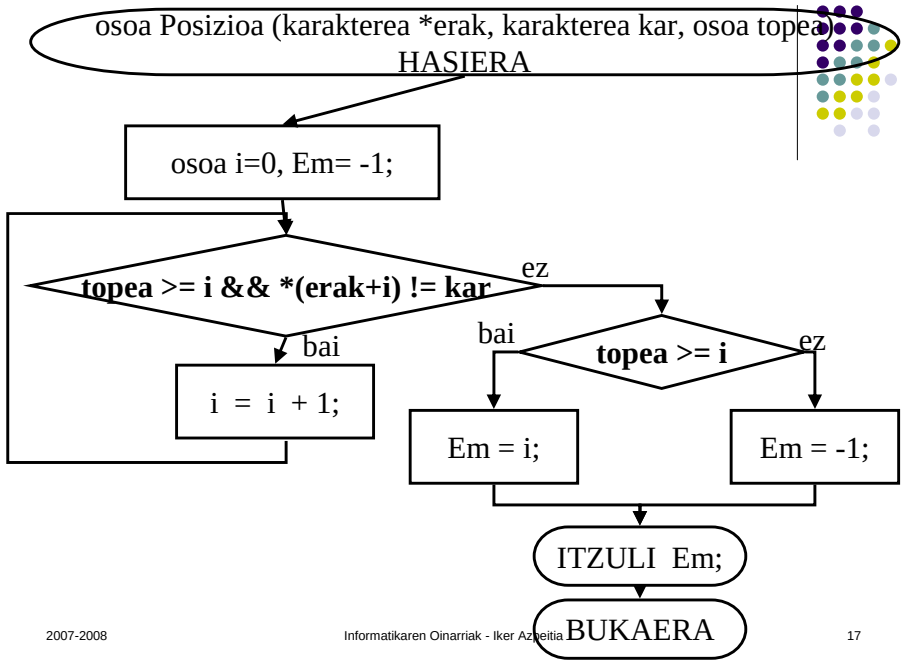
…		…
65000		KarT[0][0]
…		…
65000+n-1		KarT[0][n-1]
65000+n		KarT[1][0]
…		…
…		…
65000+(m-1) * n		KarT[m-1][0]
…		…
65000+(m*n)-1		KarT[m-1][n-1]
…		

3.1 ADIBIDEAK
Posizioa azpi-algoritmoa

- Zehaztapena:
  - Aurrebaldintza:
    - Zer da:
      - **erak: karaktere kateari erakuslea**
      - **kar: aurkitu beharreko karakterea**
      - **topea: gehienez zenbat posizio bilatu**
    - Mota:
      - **karakterea *erak;**
      - **karakterea kar;**
      - **osoa topea; topea $\in$ Z**
    - Baldintza indibidualak: t**opea>=0**
    - Erlazioak:

3.2 ADIBIDEAK
Posizioa azpi-algoritmoa

- Zehaztapena:
  - Ondorengo baldintza:
    - Zer da: **Em: zein posiziotan dagoen kar *erak apuntatzen duen karaktere katean**
    - Mota: **Em $\in$ Z**
    - Baldintza indibidualak: **Em>=-1**
    - Erlazioak:
  - **Existitzen bada i balioa betetzen duena *(erak+i) = = kar eta i >= 0 eta i <= Topea eta beste edozein j balioak betetzen badu 0 <= j < i eta *(erak+i) ! = kar) ➔ Em = = i**
  - **Bestela ➔ Em = = -1**


osoa Posizioa (karakterea *erak, karakterea kar, osoa topea)
HASIERA
osoa i=0, Em= -1;
topea >= i && *(erak+i) != kar
ez
bai
i = i + 1;
bai
topea >= i
ez
Em = i;
Em = -1;
ITZULI Em;
BUKAERA