Pràctica 2

# Pràctica 2.1

## Fase 1:

### Disseny

Per a dissenyar la nostra pràctica hem modificat el codi proporcionat pels professors de l’assignatura de tal manera que tinguem 2 funcions separades:

*mou\_menjacocos()*

*mou\_fantasma()*

La nostra és idea és crear un thread que controli el menjacocos i un altre per cada fantasma. Aquests threads aniran controlant els seus elements fins que s’aturi la execució del joc.

El joc es pot aturar quan algun fantasma menja el menjacocos o quan l’usuari ha aconseguit menjar tots els cocos, això es controla amb les variables *fi1* i *fi2*, quan alguna d’aquestes variables pren el valor de cert s’aturen tots els threads.

Per acabar hem plantejat el codi principal de tal manera que simplement creï els threads i esperi a que finalitzin per després analitzar el resultat que han retornat.

### Implementació

A continuació s’adjunta la solució proposada:

#include <stdio.h> /\* incloure definicions de funcions estandard \*/

#include <stdlib.h> /\* per exit() \*/

#include <unistd.h> /\* per getpid() \*/

#include "winsuport.h" /\* incloure definicions de funcions propies \*/

#include <pthread.h>

#define MIN\_FIL 7 /\* definir limits de variables globals \*/

#define MAX\_FIL 25

#define MIN\_COL 10

#define MAX\_COL 80

#define MAX\_THREADS 10

#define MAX\_FANTASMES 100

/\* definir estructures d'informacio \*/

typedef struct { /\* per un objecte (menjacocos o fantasma) \*/

int f; /\* posicio actual: fila \*/

int c; /\* posicio actual: columna \*/

int d; /\* direccio actual: [0..3] \*/

float r; /\* per indicar un retard relati \*/

char a; /\* caracter anterior en pos. actual \*/

} objecte;

/\* variables globals \*/

int n\_fil1, n\_col; /\* dimensions del camp de joc \*/

char tauler[70]; /\* nom del fitxer amb el laberint de joc 70\*/

char c\_req; /\* caracter de pared del laberint \*/

pthread\_t tid[MAX\_THREADS];

pthread\_t coco;

objecte mc; /\* informacio del menjacocos \*/

objecte fantasmes[10]; /\* informacio del fantasma 1 \*/

int n\_fantasmes = 0;

int df[] = {-1, 0, 1, 0}; /\* moviments de les 4 direccions possibles \*/

int dc[] = {0, -1, 0, 1}; /\* dalt, esquerra, baix, dreta \*/

int cocos; /\* numero restant de cocos per menjar \*/

int retard; /\* valor del retard de moviment, en mil.lisegons \*/

int fi1=0, fi2=0;

/\* funcio per realitzar la carrega dels parametres de joc emmagatzemats \*/

/\* dins d'un fitxer de text, el nom del qual es passa per referencia a \*/

/\* 'nom\_fit'; si es detecta algun problema, la funcio avorta l'execucio \*/

/\* enviant un missatge per la sortida d'error i retornant el codi per- \*/

/\* tinent al SO (segons comentaris al principi del programa). \*/

void carrega\_parametres(const char \*nom\_fit)

{

FILE \*fit;

fit = fopen(nom\_fit,"rt"); /\* intenta obrir fitxer \*/

if (fit == NULL)

{ fprintf(stderr,"No s'ha pogut obrir el fitxer \'%s\'\n",nom\_fit);

exit(2);

}

if (!feof(fit)) fscanf(fit,"%d %d %s %c\n",&n\_fil1,&n\_col,tauler,&c\_req);

else {

fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom\_fit);

fclose(fit);

exit(2);

}

if ((n\_fil1 < MIN\_FIL) || (n\_fil1 > MAX\_FIL) ||

(n\_col < MIN\_COL) || (n\_col > MAX\_COL))

{

fprintf(stderr,"Error: dimensions del camp de joc incorrectes:\n");

fprintf(stderr,"\t%d =< n\_fil1 (%d) =< %d\n",MIN\_FIL,n\_fil1,MAX\_FIL);

fprintf(stderr,"\t%d =< n\_col (%d) =< %d\n",MIN\_COL,n\_col,MAX\_COL);

fclose(fit);

exit(3);

}

if (!feof(fit)) fscanf(fit,"%d %d %d %f\n",&mc.f,&mc.c,&mc.d,&mc.r);

else {

fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom\_fit);

fclose(fit);

exit(2);

}

if ((mc.f < 1) || (mc.f > n\_fil1-3) ||

(mc.c < 1) || (mc.c > n\_col-2) ||

(mc.d < 0) || (mc.d > 3))

{

fprintf(stderr,"Error: parametres menjacocos incorrectes:\n");

fprintf(stderr,"\t1 =< mc.f (%d) =< n\_fil1-3 (%d)\n",mc.f,(n\_fil1-3));

fprintf(stderr,"\t1 =< mc.c (%d) =< n\_col-2 (%d)\n",mc.c,(n\_col-2));

fprintf(stderr,"\t0 =< mc.d (%d) =< 3\n",mc.d);

fclose(fit);

exit(4);

}

int i = 0;

while(!feof(fit)){

fscanf(fit,"%d %d %d %f\n",&fantasmes[i].f,&fantasmes[i].c,&fantasmes[i].d,&fantasmes[i].r);

if ((fantasmes[i].f < 1) || (fantasmes[i].f > n\_fil1-3) ||

(fantasmes[i].c < 1) || (fantasmes[i].c > n\_col-2) ||

(fantasmes[i].d < 0) || (fantasmes[i].d > 3))

{

fprintf(stderr,"Error: parametres fantasma 1 incorrectes:\n");

fprintf(stderr,"\t1 =< f1.f (%d) =< n\_fil1-3 (%d)\n",fantasmes[i].f,(n\_fil1-3));

fprintf(stderr,"\t1 =< f1.c (%d) =< n\_col-2 (%d)\n",fantasmes[i].c,(n\_col-2));

fprintf(stderr,"\t0 =< f1.d (%d) =< 3\n",fantasmes[i].d);

fclose(fit);

exit(5);

}else{

fprintf(stderr,"Fantasma %d: Fila %d i Columna %d\n",i,fantasmes[i].f,fantasmes[i].c);

}

i++;

}

n\_fantasmes = i;

fprintf(stderr,"El valor del numero de fantasmes és %d\n",i);

fclose(fit); /\* fitxer carregat: tot OK! \*/

printf("Joc del MenjaCocos\n\tTecles: \'%c\', \'%c\', \'%c\', \'%c\', RETURN-> sortir\n",

TEC\_AMUNT, TEC\_AVALL, TEC\_DRETA, TEC\_ESQUER);

printf("prem una tecla per continuar:\n");

getchar();

}

/\* funcio per moure un fantasma una posicio; retorna 1 si el fantasma \*/

/\* captura al menjacocos, 0 altrament \*/

//int mou\_fantasma(objecte f1)

void \* mou\_fantasma(void \*index)

{

int i = (intptr\_t) index;

objecte f1 = fantasmes[i];

if(i < 0){

fprintf(stderr,"Error: index incorrecte\n");

exit(6);

}

objecte seg;

int ret;

int k, vk, nd, vd[3];

ret = 0;

do{

nd = 0; //numero de direccions disponibles

for (k=-1; k<=1; k++) /\* provar direccio actual i dir. veines \*/

{

vk = (f1.d + k) % 4; /\* direccio veina \*/

if (vk < 0) vk += 4; /\* corregeix negatius \*/

seg.f = f1.f + df[vk]; /\* calcular posicio en la nova dir.\*/

seg.c = f1.c + dc[vk];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0'))

{

vd[nd] = vk; /\* memoritza com a direccio possible \*/

nd++;

}

}

if (nd == 0){ /\* si no pot continuar, \*/

f1.d = (f1.d + 2) % 4; /\* canvia totalment de sentit \*/

}else

{

if (nd == 1){ /\* si nomes pot en una direccio \*/

f1.d = vd[0]; /\* li assigna aquesta \*/

}

else{ /\* altrament \*/

f1.d = vd[rand() % nd]; /\* segueix una dir. aleatoria \*/

}

seg.f = f1.f + df[f1.d]; /\* calcular seguent posicio final \*/

seg.c = f1.c + dc[f1.d];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

win\_escricar(f1.f,f1.c,f1.a,NO\_INV); /\* esborra posicio anterior \*/

f1.f = seg.f;

f1.c = seg.c;

f1.a = seg.a; /\* actualitza posicio \*/

win\_escricar(f1.f,f1.c,'1'+i,NO\_INV); /\* redibuixa fantasma \*/

if (f1.a == '0') ret = 1; /\* ha capturat menjacocos \*/

}

fi2=ret;

win\_retard(mc.r\*retard);

}while(!fi1 && !fi2);

return ((void\*) (intptr\_t) fi2);

}

/\* funcio per moure el menjacocos una posicio, en funcio de la direccio de \*/

/\* moviment actual; retorna -1 si s'ha premut RETURN, 1 si s'ha menjat tots \*/

/\* els cocos, i 0 altrament \*/

//int mou\_menjacocos(void)

void \* mou\_menjacocos(void \* null)

{

char strin[12];

objecte seg;

int tec, ret;

ret = 0;

do{

tec = win\_gettec();

if (tec != 0)

switch (tec) /\* modificar direccio menjacocos segons tecla \*/

{

case TEC\_AMUNT: mc.d = 0; break;

case TEC\_ESQUER: mc.d = 1; break;

case TEC\_AVALL: mc.d = 2; break;

case TEC\_DRETA: mc.d = 3; break;

case TEC\_RETURN: ret = -1; break;

}

seg.f = mc.f + df[mc.d]; /\* calcular seguent posicio \*/

seg.c = mc.c + dc[mc.d];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

if ((seg.a == ' ') || (seg.a == '.'))

{

win\_escricar(mc.f,mc.c,' ',NO\_INV); /\* esborra posicio anterior \*/

mc.f = seg.f; mc.c = seg.c; /\* actualitza posicio \*/

win\_escricar(mc.f,mc.c,'0',NO\_INV); /\* redibuixa menjacocos \*/

if (seg.a == '.')

{

cocos--;

sprintf(strin,"Cocos: %d", cocos); win\_escristr(strin);

//sprintf(strin,"Fanta: %d",n\_fantasmes); win\_escristr(strin);

if (cocos == 0) ret = 1;

}

}

fi1=ret;

win\_retard(mc.r\*retard);

}while(!fi1 && !fi2);

return ((void\*) (intptr\_t) fi1);

}

/\* funcio per inicialitar les variables i visualitzar l'estat inicial del joc \*/

void inicialitza\_joc(void)

{

int r,i,j;

char strin[12];

int ex\_code = 0;

r = win\_carregatauler(tauler,n\_fil1-1,n\_col,c\_req);

if (r == 0)

{

mc.a = win\_quincar(mc.f,mc.c);

if (mc.a == c\_req) r = -6; /\* error: menjacocos sobre pared \*/

else

{

for(int i = 0; i < n\_fantasmes; i++){

fantasmes[i].a = win\_quincar(fantasmes[i].f,fantasmes[i].c);

if (fantasmes[i].a == c\_req){

r = -7; /\* error: fantasma sobre pared \*/

ex\_code = i;

}

win\_escricar(fantasmes[i].f,fantasmes[i].c,(char) ('1'+i),NO\_INV);

}

}

}

if(r==0){

cocos = 0; /\* compta el numero total de cocos \*/

for (i=0; i<n\_fil1-1; i++)

for (j=0; j<n\_col; j++){

if (win\_quincar(i,j)=='.') cocos++;

}

win\_escricar(mc.f,mc.c,'0',NO\_INV);

if (mc.a == '.') cocos--; /\* menja primer coco \*/

sprintf(strin,"Cocos: %d", cocos);

win\_escristr(strin);

}

if (r != 0)

{ win\_fi();

fprintf(stderr,"Error: no s'ha pogut inicialitzar el joc:\n");

switch (r)

{ case -1: fprintf(stderr," nom de fitxer erroni\n"); break;

case -2: fprintf(stderr," numero de columnes d'alguna fila no coincideix amb l'amplada del tauler de joc\n"); break;

case -3: fprintf(stderr," numero de columnes del laberint incorrecte\n"); break;

case -4: fprintf(stderr," numero de files del laberint incorrecte\n"); break;

case -5: fprintf(stderr," finestra de camp de joc no oberta\n"); break;

case -6: fprintf(stderr," posicio inicial del menjacocos damunt la pared del laberint\n"); break;

case -7: fprintf(stderr," posicio inicial del fantasma %d damunt la pared del laberint\n",ex\_code); break;

}

exit(7);

}

}

void crear\_threads(){

int i;

if(pthread\_create(&coco,NULL,mou\_menjacocos,NULL)!=0){

fprintf(stderr,"Error al crear el thread del menjacocos\n");

exit(1);

}

for(i = 0; i < n\_fantasmes; i++){

if(i<MAX\_FANTASMES){

if(pthread\_create(&tid[i],NULL,mou\_fantasma,(void \*)(intptr\_t) i)!=0){

fprintf(stderr,"Error al crear el thread %d\n",i);

exit(1);

}

}

}

}

/\* programa principal \*/

int main(int n\_args, const char \*ll\_args[])

{

int rc; /\* variables locals \*/

srand(getpid()); /\* inicialitza numeros aleatoris \*/

if ((n\_args != 2) && (n\_args !=3))

{ fprintf(stderr,"Comanda: cocos0 fit\_param [retard]\n");

exit(1);

}

carrega\_parametres(ll\_args[1]);

//fprintf(stderr,"Fantasma %d: Fila %d i Columna %d\n",1,fantasmes[1].f,fantasmes[1].c);

if (n\_args == 3) retard = atoi(ll\_args[2]);

else retard = 100;

rc = win\_ini(&n\_fil1,&n\_col,'+',INVERS); /\* intenta crear taulell \*/

if (rc == 0) /\* si aconsegueix accedir a l'entorn CURSES \*/

{

inicialitza\_joc();

crear\_threads(); //creem els threads

//fi1 = 0; fi2 = 0;

/\*do{

win\_retard(retard);

}while(!fi1 && !fi2);\*/

for(int th=0; th<n\_fantasmes;th++){

pthread\_join(tid[th], (void \*\*)&fi2);

}

pthread\_join(coco, (void \*\*)&fi1);

win\_fi();

if (fi1 == -1) printf("S'ha aturat el joc amb tecla RETURN!\n");

else { if (fi1) printf("Ha guanyat l'usuari!\n");

else printf("Ha guanyat l'ordinador!\n"); }

}

else

{ fprintf(stderr,"Error: no s'ha pogut crear el taulell:\n");

switch (rc)

{ case -1: fprintf(stderr,"camp de joc ja creat!\n");

break;

case -2: fprintf(stderr,"no s'ha pogut inicialitzar l'entorn de curses!\n");

break;

case -3: fprintf(stderr,"les mides del camp demanades son massa grans!\n");

break;

case -4: fprintf(stderr,"no s'ha pogut crear la finestra!\n");

break;

}

exit(6);

}

return(0);

}

### Joc de Proves

Per a comprovar el funcionament del nostre codi s’han fet els següents jocs de proves:

|  |  |  |
| --- | --- | --- |
| **Especificació:** | **Paràmetres** | **Resultat esperat** |
| Prova d’un tauler buit sense fantasmes, només amb el menjacocos. | * prova1.txt | El jugador pot moure el menjacocos pel mapa, no apareix cap fantasma.  El jugador guanya quan ha menjat tots els cocos. |
| Prova d’un tauler amb els següents paràmetres incorrectes:   * Direcció fantasma 1 fora de rang | * prova2.txt | El programa mostra un error i no es pot jugar. |
| Prova d’un tauler amb els següents paràmetres incorrectes:   * Fila fantasma 1 fora de rang | * prova3.txt | El programa mostra un error i no es pot jugar. |
| Prova de posar un fantasma sobre la paret. | * prova4.txt | El programa mostra un error i no es pot jugar. |
| Prova d’un tauler amb un nombre de fantasmes més gran que MAX\_FANTASMES. | * prova5.txt | No es permet jugar ja que hi ha una inconsistència en els paràmetres. |
| Prova d’un joc normal | * prova6.txt | Es permet jugar, però es noten errors de sincronització entre els threads fent que el joc sigui injugable. |

## Fase 2:

### Disseny:

Per a fer el disseny d’aquesta part hem analitzat les variables globals que es poden fer servir des de 2 threads concurrentment.

Hem identificat principalment que la secció en la qual garantir la exclusivitat d’accés a la memòria és la part de dins del bucle, així que hem implementat el següent pseudocodi:

*bloquejar()*

*do*

*{*

*[..]*

*desbloquejar()*

*// permetem al següent thread accedir a la memòria*

*bloquejar()*

*} while( .. )*

*desbloquejar()*

La nostra idea es que així tant podem permetre que els diferents threads puguin treballar concurrentment reservant la secció crítica de cada un.

Cal remarcar que aquest esquema s’ha implementat tant per moure els fantasmes com per moure el menjacocos, així garantim una integritat total al codi.

### Implementació:

S’ha realitzat la següent implementació del codi de solució:

#include <stdio.h> /\* incloure definicions de funcions estandard \*/

#include <stdlib.h> /\* per exit() \*/

#include <unistd.h> /\* per getpid() \*/

#include "winsuport.h" /\* incloure definicions de funcions propies \*/

#include <pthread.h>

#define MIN\_FIL 7 /\* definir limits de variables globals \*/

#define MAX\_FIL 25

#define MIN\_COL 10

#define MAX\_COL 80

#define MAX\_THREADS 10

#define MAX\_FANTASMES 100

/\* definir estructures d'informacio \*/

typedef struct { /\* per un objecte (menjacocos o fantasma) \*/

int f; /\* posicio actual: fila \*/

int c; /\* posicio actual: columna \*/

int d; /\* direccio actual: [0..3] \*/

float r; /\* per indicar un retard relati \*/

char a; /\* caracter anterior en pos. actual \*/

} objecte;

/\* variables globals \*/

int n\_fil1, n\_col; /\* dimensions del camp de joc \*/

char tauler[70]; /\* nom del fitxer amb el laberint de joc 70\*/

char c\_req; /\* caracter de pared del laberint \*/

pthread\_t tid[MAX\_THREADS];

pthread\_t coco;

objecte mc; /\* informacio del menjacocos \*/

objecte fantasmes[10]; /\* informacio del fantasma 1 \*/

int n\_fantasmes = 0;

int df[] = {-1, 0, 1, 0}; /\* moviments de les 4 direccions possibles \*/

int dc[] = {0, -1, 0, 1}; /\* dalt, esquerra, baix, dreta \*/

int cocos; /\* numero restant de cocos per menjar \*/

int retard; /\* valor del retard de moviment, en mil.lisegons \*/

int fi1=0, fi2=0;

pthread\_mutex\_t mutex=PTHREAD\_MUTEX\_INITIALIZER;

//pthread\_mutex\_t mutex2=PTHREAD\_MUTEX\_INITIALIZER;

/\* funcio per realitzar la carrega dels parametres de joc emmagatzemats \*/

/\* dins d'un fitxer de text, el nom del qual es passa per referencia a \*/

/\* 'nom\_fit'; si es detecta algun problema, la funcio avorta l'execucio \*/

/\* enviant un missatge per la sortida d'error i retornant el codi per- \*/

/\* tinent al SO (segons comentaris al principi del programa). \*/

void carrega\_parametres(const char \*nom\_fit)

{

FILE \*fit;

fit = fopen(nom\_fit,"rt"); /\* intenta obrir fitxer \*/

if (fit == NULL)

{ fprintf(stderr,"No s'ha pogut obrir el fitxer \'%s\'\n",nom\_fit);

exit(2);

}

if (!feof(fit)) fscanf(fit,"%d %d %s %c\n",&n\_fil1,&n\_col,tauler,&c\_req);

else {

fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom\_fit);

fclose(fit);

exit(2);

}

if ((n\_fil1 < MIN\_FIL) || (n\_fil1 > MAX\_FIL) ||

(n\_col < MIN\_COL) || (n\_col > MAX\_COL))

{

fprintf(stderr,"Error: dimensions del camp de joc incorrectes:\n");

fprintf(stderr,"\t%d =< n\_fil1 (%d) =< %d\n",MIN\_FIL,n\_fil1,MAX\_FIL);

fprintf(stderr,"\t%d =< n\_col (%d) =< %d\n",MIN\_COL,n\_col,MAX\_COL);

fclose(fit);

exit(3);

}

if (!feof(fit)) fscanf(fit,"%d %d %d %f\n",&mc.f,&mc.c,&mc.d,&mc.r);

else {

fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom\_fit);

fclose(fit);

exit(2);

}

if ((mc.f < 1) || (mc.f > n\_fil1-3) ||

(mc.c < 1) || (mc.c > n\_col-2) ||

(mc.d < 0) || (mc.d > 3))

{

fprintf(stderr,"Error: parametres menjacocos incorrectes:\n");

fprintf(stderr,"\t1 =< mc.f (%d) =< n\_fil1-3 (%d)\n",mc.f,(n\_fil1-3));

fprintf(stderr,"\t1 =< mc.c (%d) =< n\_col-2 (%d)\n",mc.c,(n\_col-2));

fprintf(stderr,"\t0 =< mc.d (%d) =< 3\n",mc.d);

fclose(fit);

exit(4);

}

int i = 0;

while(!feof(fit)){

fscanf(fit,"%d %d %d %f\n",&fantasmes[i].f,&fantasmes[i].c,&fantasmes[i].d,&fantasmes[i].r);

if ((fantasmes[i].f < 1) || (fantasmes[i].f > n\_fil1-3) ||

(fantasmes[i].c < 1) || (fantasmes[i].c > n\_col-2) ||

(fantasmes[i].d < 0) || (fantasmes[i].d > 3))

{

fprintf(stderr,"Error: parametres fantasma 1 incorrectes:\n");

fprintf(stderr,"\t1 =< f1.f (%d) =< n\_fil1-3 (%d)\n",fantasmes[i].f,(n\_fil1-3));

fprintf(stderr,"\t1 =< f1.c (%d) =< n\_col-2 (%d)\n",fantasmes[i].c,(n\_col-2));

fprintf(stderr,"\t0 =< f1.d (%d) =< 3\n",fantasmes[i].d);

fclose(fit);

exit(5);

}else{

fprintf(stderr,"Fantasma %d: Fila %d i Columna %d\n",i,fantasmes[i].f,fantasmes[i].c);

}

i++;

}

n\_fantasmes = i;

fprintf(stderr,"El valor del numero de fantasmes és %d\n",i);

fclose(fit); /\* fitxer carregat: tot OK! \*/

printf("Joc del MenjaCocos\n\tTecles: \'%c\', \'%c\', \'%c\', \'%c\', RETURN-> sortir\n",

TEC\_AMUNT, TEC\_AVALL, TEC\_DRETA, TEC\_ESQUER);

printf("prem una tecla per continuar:\n");

getchar();

}

/\* funcio per moure un fantasma una posicio; retorna 1 si el fantasma \*/

/\* captura al menjacocos, 0 altrament \*/

//int mou\_fantasma(objecte f1)

void \* mou\_fantasma(void \*index)

{

int i = (intptr\_t) index;

objecte f1 = fantasmes[i];

if(i < 0){

fprintf(stderr,"Error: index incorrecte\n");

exit(6);

}

objecte seg;

int ret;

int k, vk, nd, vd[3];

ret = 0;

pthread\_mutex\_lock(&mutex);

do{

nd = 0; //numero de direccions disponibles

for (k=-1; k<=1; k++) /\* provar direccio actual i dir. veines \*/

{

vk = (f1.d + k) % 4; /\* direccio veina \*/

if (vk < 0) vk += 4; /\* corregeix negatius \*/

seg.f = f1.f + df[vk]; /\* calcular posicio en la nova dir.\*/

seg.c = f1.c + dc[vk];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0'))

{

vd[nd] = vk; /\* memoritza com a direccio possible \*/

nd++;

}

}

if (nd == 0){ /\* si no pot continuar, \*/

f1.d = (f1.d + 2) % 4; /\* canvia totalment de sentit \*/

}else

{

if (nd == 1){ /\* si nomes pot en una direccio \*/

f1.d = vd[0]; /\* li assigna aquesta \*/

}

else{ /\* altrament \*/

f1.d = vd[rand() % nd]; /\* segueix una dir. aleatoria \*/

}

seg.f = f1.f + df[f1.d]; /\* calcular seguent posicio final \*/

seg.c = f1.c + dc[f1.d];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

win\_escricar(f1.f,f1.c,f1.a,NO\_INV); /\* esborra posicio anterior \*/

f1.f = seg.f;

f1.c = seg.c;

f1.a = seg.a; /\* actualitza posicio \*/

win\_escricar(f1.f,f1.c,'1'+i,NO\_INV); /\* redibuixa fantasma \*/

if (f1.a == '0') ret = 1; /\* ha capturat menjacocos \*/

}

fi2=ret;

pthread\_mutex\_unlock(&mutex);

win\_retard(mc.r\*retard);

pthread\_mutex\_lock(&mutex);

}while(!fi1 && !fi2);

pthread\_mutex\_unlock(&mutex);

return ((void\*) (intptr\_t) fi2);

}

/\* funcio per moure el menjacocos una posicio, en funcio de la direccio de \*/

/\* moviment actual; retorna -1 si s'ha premut RETURN, 1 si s'ha menjat tots \*/

/\* els cocos, i 0 altrament \*/

//int mou\_menjacocos(void)

void \* mou\_menjacocos(void \* null)

{

char strin[12];

objecte seg;

int tec, ret;

ret = 0;

do{

pthread\_mutex\_lock(&mutex);

tec = win\_gettec();

if (tec != 0)

switch (tec) /\* modificar direccio menjacocos segons tecla \*/

{

case TEC\_AMUNT: mc.d = 0; break;

case TEC\_ESQUER: mc.d = 1; break;

case TEC\_AVALL: mc.d = 2; break;

case TEC\_DRETA: mc.d = 3; break;

case TEC\_RETURN: ret = -1; break;

}

seg.f = mc.f + df[mc.d]; /\* calcular seguent posicio \*/

seg.c = mc.c + dc[mc.d];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

if ((seg.a == ' ') || (seg.a == '.'))

{

win\_escricar(mc.f,mc.c,' ',NO\_INV); /\* esborra posicio anterior \*/

mc.f = seg.f; mc.c = seg.c; /\* actualitza posicio \*/

win\_escricar(mc.f,mc.c,'0',NO\_INV); /\* redibuixa menjacocos \*/

if (seg.a == '.')

{

cocos--;

sprintf(strin,"Cocos: %d", cocos); win\_escristr(strin);

//sprintf(strin,"Fanta: %d",n\_fantasmes); win\_escristr(strin);

if (cocos == 0) ret = 1;

}

}

fi1=ret;

pthread\_mutex\_unlock(&mutex);

win\_retard(mc.r\*retard);

}while(!fi1 && !fi2);

return ((void\*) (intptr\_t) fi1);

}

/\* funcio per inicialitar les variables i visualitzar l'estat inicial del joc \*/

void inicialitza\_joc(void)

{

int r,i,j;

char strin[12];

int ex\_code = 0;

r = win\_carregatauler(tauler,n\_fil1-1,n\_col,c\_req);

if (r == 0)

{

mc.a = win\_quincar(mc.f,mc.c);

if (mc.a == c\_req) r = -6; /\* error: menjacocos sobre pared \*/

else

{

for(int i = 0; i < n\_fantasmes; i++){

fantasmes[i].a = win\_quincar(fantasmes[i].f,fantasmes[i].c);

if (fantasmes[i].a == c\_req){

r = -7; /\* error: fantasma sobre pared \*/

ex\_code = i;

}

win\_escricar(fantasmes[i].f,fantasmes[i].c,(char) ('1'+i),NO\_INV);

}

}

}

if(r==0){

cocos = 0; /\* compta el numero total de cocos \*/

for (i=0; i<n\_fil1-1; i++)

for (j=0; j<n\_col; j++){

pthread\_mutex\_lock(&mutex);

if (win\_quincar(i,j)=='.') cocos++;

pthread\_mutex\_unlock(&mutex);

}

pthread\_mutex\_lock(&mutex);

win\_escricar(mc.f,mc.c,'0',NO\_INV);

pthread\_mutex\_unlock(&mutex);

pthread\_mutex\_lock(&mutex);

if (mc.a == '.') cocos--; /\* menja primer coco \*/

pthread\_mutex\_unlock(&mutex);

sprintf(strin,"Cocos: %d", cocos);

pthread\_mutex\_lock(&mutex);

win\_escristr(strin);

pthread\_mutex\_unlock(&mutex);

}

if (r != 0)

{ win\_fi();

fprintf(stderr,"Error: no s'ha pogut inicialitzar el joc:\n");

switch (r)

{ case -1: fprintf(stderr," nom de fitxer erroni\n"); break;

case -2: fprintf(stderr," numero de columnes d'alguna fila no coincideix amb l'amplada del tauler de joc\n"); break;

case -3: fprintf(stderr," numero de columnes del laberint incorrecte\n"); break;

case -4: fprintf(stderr," numero de files del laberint incorrecte\n"); break;

case -5: fprintf(stderr," finestra de camp de joc no oberta\n"); break;

case -6: fprintf(stderr," posicio inicial del menjacocos damunt la pared del laberint\n"); break;

case -7: fprintf(stderr," posicio inicial del fantasma %d damunt la pared del laberint\n",ex\_code); break;

}

exit(7);

}

}

void crear\_threads(){

int i;

if(pthread\_create(&coco,NULL,mou\_menjacocos,NULL)!=0){

fprintf(stderr,"Error al crear el thread del menjacocos\n");

exit(1);

}

for(i = 0; i < n\_fantasmes; i++){

if(i<MAX\_FANTASMES){

if(pthread\_create(&tid[i],NULL,mou\_fantasma,(void \*)(intptr\_t) i)!=0){

fprintf(stderr,"Error al crear el thread %d\n",i);

exit(1);

}

}

}

}

/\* programa principal \*/

int main(int n\_args, const char \*ll\_args[])

{

int rc; /\* variables locals \*/

srand(getpid()); /\* inicialitza numeros aleatoris \*/

if ((n\_args != 2) && (n\_args !=3))

{ fprintf(stderr,"Comanda: cocos0 fit\_param [retard]\n");

exit(1);

}

carrega\_parametres(ll\_args[1]);

//fprintf(stderr,"Fantasma %d: Fila %d i Columna %d\n",1,fantasmes[1].f,fantasmes[1].c);

if (n\_args == 3) retard = atoi(ll\_args[2]);

else retard = 100;

rc = win\_ini(&n\_fil1,&n\_col,'+',INVERS); /\* intenta crear taulell \*/

if (rc == 0) /\* si aconsegueix accedir a l'entorn CURSES \*/

{

pthread\_mutex\_init(&mutex, NULL);

inicialitza\_joc();

crear\_threads(); //creem els threads

//fi1 = 0; fi2 = 0;

/\*do{

win\_retard(retard);

}while(!fi1 && !fi2);\*/

for(int th=0; th<n\_fantasmes;th++){

pthread\_join(tid[th], (void \*\*)&fi2);

}

pthread\_join(coco, (void \*\*)&fi1);

pthread\_mutex\_destroy(&mutex);

win\_fi();

if (fi1 == -1) printf("S'ha aturat el joc amb tecla RETURN!\n");

else { if (fi1) printf("Ha guanyat l'usuari!\n");

else printf("Ha guanyat l'ordinador!\n"); }

}

else

{ fprintf(stderr,"Error: no s'ha pogut crear el taulell:\n");

switch (rc)

{ case -1: fprintf(stderr,"camp de joc ja creat!\n");

break;

case -2: fprintf(stderr,"no s'ha pogut inicialitzar l'entorn de curses!\n");

break;

case -3: fprintf(stderr,"les mides del camp demanades son massa grans!\n");

break;

case -4: fprintf(stderr,"no s'ha pogut crear la finestra!\n");

break;

}

exit(6);

}

return(0);

}

### Joc de proves:

Per a comprovar el funcionament del nostre codi s’han fet els següents jocs de proves:

|  |  |  |
| --- | --- | --- |
| **Especificació:** | **Paràmetres** | **Resultat esperat** |
| Prova d’un tauler buit sense fantasmes, només amb el menjacocos. | * prova1.txt | El jugador pot moure el menjacocos pel mapa, no apareix cap fantasma.  El jugador guanya quan ha menjat tots els cocos. |
| Prova d’un tauler amb els següents paràmetres incorrectes:   * Direcció fantasma 1 fora de rang | * prova2.txt | El programa mostra un error i no es pot jugar. |
| Prova d’un tauler amb els següents paràmetres incorrectes:   * Fila fantasma 1 fora de rang | * prova3.txt | El programa mostra un error i no es pot jugar. |
| Prova de posar un fantasma sobre la paret. | * prova4.txt | El programa mostra un error i no es pot jugar. |
| Prova d’un tauler amb un nombre de fantasmes més gran que MAX\_FANTASMES. | * prova5.txt | No es permet jugar ja que hi ha una inconsistència en els paràmetres. |
| Prova d’un joc normal | * prova6.txt | Es permet jugar sense problemes i no surten errors de sincronització entre els threads. |

# Pràctica 2.2

## Fase 1:

### Disseny

Per a dissenyar la nostra solució primer hem agafat el procediment que controla el moviment dels fantasmes i l’hem posat en un fitxer a part, anomenat *fantasmes.c*.

Seguidament hem identificat les variables següents que són les que desde aquest procediment es fan servir i alhora desde el codi principal i el moviment del fantasma també:

|  |  |
| --- | --- |
| fi1 | Variable que indica si el menjacocos ha finalitzat de menjar tots els cocos. |
| fi2 | Variable que indica si algun fantasma ha menjat el menjacocos. |
| fantasmes | Taula d’objectes que conté els atributs de cada fantasma. |
| mc | Objecte que conté els atributs del menjacocos |
| retard | Paràmetre que indica el retard a contemplar en el moviment. |
| id\_win | Id de la finestra on s’ha iniciat el joc. |

Per a permetre un correcte funcionament de la pràctica i una correcta sincronització de les dades s’ha hagut de transferir la informació anteriorment comentada a cada procés mitjançant la rutina exec(). A més dels paràmetres anteriors també s’ha inclòs:

* Identificador del número de fantasma.
* Número de files del tauler.
* Número de columnes del tauler.

Un cop creat els processos fills hem fet que el codi principal es dediqui principalment a actualitzar la pantalla mentre no s’acabi el joc.

Cal remarcar que el moviment del fantasma s’ha mantingut en un thread tal i com s’ha fet en la Fase 1.

### Implementació:

*Cocos3.c:*

/\*\*

\* @file cocos3.c

\* @brief Codi que controla el moviment d'un joc menjacocos sense sincronització

\* @author Assmaa Ouladali i David Domènech

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* \*/

/\* cocos3.c \*/

/\* \*/

/\* Programa inicial d'exemple per a les practiques 2.1 i 2.2 de FSO. \*/

/\* Es tracta del joc del menjacocos: es dibuixa un laberint amb una \*/

/\* serie de punts (cocos), els quals han de ser "menjats" pel menja- \*/

/\* cocos. Aquest menjacocos es representara amb el caracter '0', i el \*/

/\* moura l'usuari amb les tecles 'w' (adalt), 's' (abaix), 'd' (dreta) \*/

/\* i 'a' (esquerra). Simultaniament hi haura un conjunt de fantasmes, \*/

/\* representats per numeros de l'1 al 9, que intentaran capturar al \*/

/\* menjacocos. En la primera versio del programa, nomes hi ha un fan- \*/

/\* tasma. \*/

/\* Evidentment, es tracta de menjar tots els punts abans que algun fan- \*/

/\* tasma atrapi al menjacocos. \*/

/\* \*/

/\* Arguments del programa: \*/

/\* per controlar la posicio de tots els elements del joc, cal indicar \*/

/\* el nom d'un fitxer de text que contindra la seguent informacio: \*/

/\* n\_fil1 n\_col fit\_tauler creq \*/

/\* mc\_f mc\_c mc\_d mc\_r \*/

/\* f1\_f f1\_c f1\_d f1\_r \*/

/\* \*/

/\* on 'n\_fil1', 'n\_col' son les dimensions del taulell de joc, mes una \*/

/\* fila pels missatges de text a l'ultima linia. "fit\_tauler" es el nom \*/

/\* d'un fitxer de text que contindra el dibuix del laberint, amb num. de \*/

/\* files igual a 'n\_fil1'-1 i num. de columnes igual a 'n\_col'. Dins \*/

/\* d'aquest fitxer, hi hauran caracter ASCCII que es representaran en \*/

/\* pantalla tal qual, excepte el caracters iguals a 'creq', que es visua-\*/

/\* litzaran invertits per representar la paret. \*/

/\* Els parametres 'mc\_f', 'mc\_c' indiquen la posicio inicial de fila i \*/

/\* columna del menjacocos, aixi com la direccio inicial de moviment \*/

/\* (0 -> amunt, 1-> esquerra, 2-> avall, 3-> dreta). Els parametres \*/

/\* 'f1\_f', 'f1\_c' i 'f1\_d' corresponen a la mateixa informacio per al \*/

/\* fantasma 1. El programa verifica que la primera posicio del menja- \*/

/\* cocos o del fantasma no coincideixi amb un bloc de paret del laberint.\*/

/\* 'mc\_r' 'f1\_r' son dos reals que multipliquen el retard del moviment. \*/

/\* A mes, es podra afegir un segon argument opcional per indicar el \*/

/\* retard de moviment del menjacocos i dels fantasmes (en ms); \*/

/\* el valor per defecte d'aquest parametre es 100 (1 decima de segon). \*/

/\* \*/

/\* Compilar i executar: \*/

/\* El programa invoca les funcions definides a 'winsuport.h', les \*/

/\* quals proporcionen una interficie senzilla per crear una finestra \*/

/\* de text on es poden escriure caracters en posicions especifiques de \*/

/\* la pantalla (basada en CURSES); per tant, el programa necessita ser \*/

/\* compilat amb la llibreria 'curses': \*/

/\* \*/

/\* $ gcc -Wall cocos0.c winsuport.o -o cocos0 -lcurses \*/

/\* $ ./cocos0 fit\_param [retard] \*/

/\* \*/

/\* Codis de retorn: \*/

/\* El programa retorna algun dels seguents codis al SO: \*/

/\* 0 ==> funcionament normal \*/

/\* 1 ==> numero d'arguments incorrecte \*/

/\* 2 ==> fitxer de configuracio no accessible \*/

/\* 3 ==> dimensions del taulell incorrectes \*/

/\* 4 ==> parametres del menjacocos incorrectes \*/

/\* 5 ==> parametres d'algun fantasma incorrectes \*/

/\* 6 ==> no s'ha pogut crear el camp de joc \*/

/\* 7 ==> no s'ha pogut inicialitzar el joc \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h> /\* incloure definicions de funcions estandard \*/

#include <stdlib.h> /\* per exit() \*/

#include <unistd.h> /\* per getpid() \*/

#include "winsuport2.h" /\* incloure definicions de funcions propies \*/

#include <pthread.h>

#include "memoria.h"

#include <sys/wait.h>

#define MIN\_FIL 7 /\* definir limits de variables globals \*/

#define MAX\_FIL 25

#define MIN\_COL 10

#define MAX\_COL 80

#define MAX\_PROCS 10

#define MAX\_FANTASMES 100

/\* definir estructures d'informacio \*/

typedef struct { /\* per un objecte (menjacocos o fantasma) \*/

int f; /\* posicio actual: fila \*/

int c; /\* posicio actual: columna \*/

int d; /\* direccio actual: [0..3] \*/

float r; /\* per indicar un retard relati \*/

char a; /\* caracter anterior en pos. actual \*/

} objecte;

int df[] = {-1, 0, 1, 0}; /\* moviments de les 4 direccions possibles \*/

int dc[] = {0, -1, 0, 1}; /\* dalt, esquerra, baix, dreta \*/

/\* variables globals \*/

int n\_fil1, n\_col; /\* dimensions del camp de joc \*/

char tauler[70]; /\* nom del fitxer amb el laberint de joc 70\*/

char c\_req; /\* caracter de pared del laberint \*/

pid\_t tpid[MAX\_PROCS];

pthread\_t coco;

objecte mc; /\* informacio del menjacocos \*/

objecte fantasmes[10]; /\* informacio del fantasma 1 \*/

int n\_fantasmes = 0;

int cocos; /\* numero restant de cocos per menjar \*/

int retard; /\* valor del retard de moviment, en mil.lisegons \*/

int fi1=0, fi2=0;

/\* funcio per realitzar la carrega dels parametres de joc emmagatzemats \*/

/\* dins d'un fitxer de text, el nom del qual es passa per referencia a \*/

/\* 'nom\_fit'; si es detecta algun problema, la funcio avorta l'execucio \*/

/\* enviant un missatge per la sortida d'error i retornant el codi per- \*/

/\* tinent al SO (segons comentaris al principi del programa). \*/

void carrega\_parametres(const char \*nom\_fit)

{

FILE \*fit;

fit = fopen(nom\_fit,"rt"); /\* intenta obrir fitxer \*/

if (fit == NULL)

{ fprintf(stderr,"No s'ha pogut obrir el fitxer \'%s\'\n",nom\_fit);

exit(2);

}

if (!feof(fit)){

fscanf(fit,"%d %d %s %c\n",&n\_fil1,&n\_col,tauler,&c\_req);

fprintf(stderr,"\nHa carregat el tauler\n'\n");

}

else {

fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom\_fit);

fclose(fit);

exit(2);

}

if ((n\_fil1 < MIN\_FIL) || (n\_fil1 > MAX\_FIL) ||

(n\_col < MIN\_COL) || (n\_col > MAX\_COL))

{

fprintf(stderr,"Error: dimensions del camp de joc incorrectes:\n");

fprintf(stderr,"\t%d =< n\_fil1 (%d) =< %d\n",MIN\_FIL,n\_fil1,MAX\_FIL);

fprintf(stderr,"\t%d =< n\_col (%d) =< %d\n",MIN\_COL,n\_col,MAX\_COL);

fclose(fit);

exit(3);

}

if (!feof(fit)) fscanf(fit,"%d %d %d %f\n",&mc.f,&mc.c,&mc.d,&mc.r);

else {

fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom\_fit);

fclose(fit);

exit(2);

}

if ((mc.f < 1) || (mc.f > n\_fil1-3) ||

(mc.c < 1) || (mc.c > n\_col-2) ||

(mc.d < 0) || (mc.d > 3))

{

fprintf(stderr,"Error: parametres menjacocos incorrectes:\n");

fprintf(stderr,"\t1 =< mc.f (%d) =< n\_fil1-3 (%d)\n",mc.f,(n\_fil1-3));

fprintf(stderr,"\t1 =< mc.c (%d) =< n\_col-2 (%d)\n",mc.c,(n\_col-2));

fprintf(stderr,"\t0 =< mc.d (%d) =< 3\n",mc.d);

fclose(fit);

exit(4);

}

int i = 0;

while(!feof(fit)){

fscanf(fit,"%d %d %d %f\n",&fantasmes[i].f,&fantasmes[i].c,&fantasmes[i].d,&fantasmes[i].r);

if ((fantasmes[i].f < 1) || (fantasmes[i].f > n\_fil1-3) ||

(fantasmes[i].c < 1) || (fantasmes[i].c > n\_col-2) ||

(fantasmes[i].d < 0) || (fantasmes[i].d > 3))

{

fprintf(stderr,"Error: parametres fantasma 1 incorrectes:\n");

fprintf(stderr,"\t1 =< f1.f (%d) =< n\_fil1-3 (%d)\n",fantasmes[i].f,(n\_fil1-3));

fprintf(stderr,"\t1 =< f1.c (%d) =< n\_col-2 (%d)\n",fantasmes[i].c,(n\_col-2));

fprintf(stderr,"\t0 =< f1.d (%d) =< 3\n",fantasmes[i].d);

fclose(fit);

exit(5);

}else{

fprintf(stderr,"Fantasma %d: Fila %d i Columna %d\n",i,fantasmes[i].f,fantasmes[i].c);

}

i++;

}

n\_fantasmes = i;

fprintf(stderr,"El valor del numero de fantasmes és %d\n",i);

fclose(fit); /\* fitxer carregat: tot OK! \*/

printf("Joc del MenjaCocos\n\tTecles: \'%c\', \'%c\', \'%c\', \'%c\', RETURN-> sortir\n",

TEC\_AMUNT, TEC\_AVALL, TEC\_DRETA, TEC\_ESQUER);

printf("prem una tecla per continuar:\n");

getchar();

}

/\* funcio per moure el menjacocos una posicio, en funcio de la direccio de \*/

/\* moviment actual; retorna -1 si s'ha premut RETURN, 1 si s'ha menjat tots \*/

/\* els cocos, i 0 altrament \*/

//int mou\_menjacocos(void)

void \* mou\_menjacocos(void \* null)

{

char strin[12];

objecte seg;

int tec, ret;

ret = 0;

do{

tec = win\_gettec();

//win\_update();

if (tec != 0)

switch (tec) /\* modificar direccio menjacocos segons tecla \*/

{

case TEC\_AMUNT: mc.d = 0; break;

case TEC\_ESQUER: mc.d = 1; break;

case TEC\_AVALL: mc.d = 2; break;

case TEC\_DRETA: mc.d = 3; break;

case TEC\_RETURN: ret = -1; break;

}

seg.f = mc.f + df[mc.d]; /\* calcular seguent posicio \*/

seg.c = mc.c + dc[mc.d];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

if ((seg.a == ' ') || (seg.a == '.'))

{

win\_escricar(mc.f,mc.c,' ',NO\_INV); /\* esborra posicio anterior \*/

mc.f = seg.f; mc.c = seg.c; /\* actualitza posicio \*/

win\_escricar(mc.f,mc.c,'0',NO\_INV); /\* redibuixa menjacocos \*/

if (seg.a == '.')

{

cocos--;

fprintf(stderr,"cocos = %d\n",cocos);

sprintf(strin,"Cocos: %d", cocos); win\_escristr(strin);

//sprintf(strin,"Fanta: %d",n\_fantasmes); win\_escristr(strin);

if (cocos == 0) ret = 1;

}

}

fi1=ret;

win\_retard(mc.r\*retard);

}while(!fi1 && !fi2);

return ((void\*) (intptr\_t) fi1);

}

/\* funcio per inicialitar les variables i visualitzar l'estat inicial del joc \*/

void inicialitza\_joc(void)

{

int r,i,j;

char strin[12];

int ex\_code = 0;

fprintf(stderr,"n\_files = %d, n\_cols = %d",n\_fil1, n\_col);

for(int i=0;i<4;i++){

fprintf(stderr,"%c", tauler[i]);

}

r = win\_carregatauler(tauler,n\_fil1-1,n\_col,c\_req);

if (r == 0)

{

mc.a = win\_quincar(mc.f,mc.c);

if (mc.a == c\_req) r = -6; /\* error: menjacocos sobre pared \*/

else

{

for(int i = 0; i < n\_fantasmes; i++){

fantasmes[i].a = win\_quincar(fantasmes[i].f,fantasmes[i].c);

if (fantasmes[i].a == c\_req){

r = -7; /\* error: fantasma sobre pared \*/

ex\_code = i;

}

win\_escricar(fantasmes[i].f,fantasmes[i].c,(char) ('1'+i),NO\_INV);

}

}

}

if(r ==0){

cocos = 0; /\* compta el numero total de cocos \*/

for (i=0; i<n\_fil1-1; i++)

for (j=0; j<n\_col; j++){

if (win\_quincar(i,j)=='.') cocos++;

}

fprintf(stderr,"\nAL peincipi hi han cocos = %d\n",cocos);

win\_escricar(mc.f,mc.c,'0',NO\_INV);

if (mc.a == '.') cocos--; /\* menja primer coco \*/

fprintf(stderr,"Ara passen a cocos = %d\n",cocos);

sprintf(strin,"Cocos: %d", cocos);

win\_escristr(strin);

}

if (r < 0)

{ win\_fi();

fprintf(stderr,"Error: no s'ha pogut inicialitzar el joc:\n");

switch (r)

{ case -1: fprintf(stderr," nom de fitxer erroni\n"); break;

case -2: fprintf(stderr," numero de columnes d'alguna fila no coincideix amb l'amplada del tauler de joc\n"); break;

case -3: fprintf(stderr," numero de columnes del laberint incorrecte\n"); break;

case -4: fprintf(stderr," numero de files del laberint incorrecte\n"); break;

case -5: fprintf(stderr," finestra de camp de joc no oberta\n"); break;

case -6: fprintf(stderr," posicio inicial del menjacocos damunt la pared del laberint\n"); break;

case -7: fprintf(stderr," posicio inicial del fantasma %d damunt la pared del laberint\n",ex\_code); break;

}

exit(7);

}

}

void actualitza\_pantalla(){

char strin[12];

win\_carregatauler(tauler,n\_fil1-1,n\_col,c\_req);

for (int i=0; i<n\_fil1-1; i++)

for (int j=0; j<n\_col; j++){

if (win\_quincar(i,j)=='.') cocos++;

}

for(int i = 0; i < n\_fantasmes; i++){

win\_escricar(fantasmes[i].f,fantasmes[i].c,(char) ('1'+i),NO\_INV);

}

win\_escricar(mc.f,mc.c,'0',NO\_INV);

}

/\* programa principal \*/

int main(int n\_args, const char \*ll\_args[])

{

int rc; /\* variables locals \*/

void \*p\_win;

char str\_win[20],str\_n\_fil[10],str\_n\_col[10];

//int n\_fil, n\_col;

srand(getpid()); /\* inicialitza numeros aleatoris \*/

if ((n\_args != 2) && (n\_args !=3))

{ fprintf(stderr,"Comanda: cocos0 fit\_param [retard]\n");

exit(1);

}

carrega\_parametres(ll\_args[1]);

//fprintf(stderr,"Fantasma %d: Fila %d i Columna %d\n",1,fantasmes[1].f,fantasmes[1].c);

if (n\_args == 3) retard = atoi(ll\_args[2]);

else retard = 100;

//inicialitza\_joc();

fprintf(stderr,"n\_files = %d, n\_cols = %d",n\_fil1, n\_col);

rc = win\_ini(&n\_fil1,&n\_col,'+',INVERS); /\* intenta crear taulell \*/

if (rc >= 0) /\* si aconsegueix accedir a l'entorn CURSES \*/

{

int id\_win = ini\_mem(rc); /\* crear zona mem. compartida \*/

p\_win = map\_mem(id\_win); /\* obtenir adres. de mem. compartida \*/

sprintf(str\_win,"%i",id\_win);

sprintf(str\_n\_fil,"%i",n\_fil1); /\* convertir mides camp en string \*/

sprintf(str\_n\_col,"%i",n\_col);

win\_set(p\_win,n\_fil1,n\_col); /\* crea acces a finestra oberta \*/

fprintf(stderr,"n\_files = %d, n\_cols = %d",n\_fil1, n\_col);

inicialitza\_joc();

//win\_set(p\_win,n\_fil1,n\_col);

/\*\*

\* CREEM THREAD I PROCESSOS

\*/

int i;

char str\_fi1[20];

int id\_fi1 = ini\_mem(sizeof(int)); //creem la zona de memoria compartida

int \*p\_fi1 = map\_mem(id\_fi1); //fem el mapeig de la zona de memoria compartida

\*p\_fi1 = fi1; //inicialitzem la zona de memoria compartida

sprintf(str\_fi1,"%i",id\_fi1); //passem l'identificador de la zona de memoria compartida a un string

//fem el mateix per fi2

char str\_fi2[20];

int id\_fi2 = ini\_mem(sizeof(int)); //creem la zona de memoria compartida

int \*p\_fi2 = map\_mem(id\_fi2); //fem el mapeig de la zona de memoria compartida

//\*p\_fi2 = fi2; //inicialitzem la zona de memoria compartida

sprintf(str\_fi2,"%i",id\_fi2); //passem l'identificador de la zona de memoria compartida a un string

// fem el mateix per df

char str\_df[20];

int id\_df = ini\_mem(sizeof(int)\*4); //creem la zona de memoria compartida

int \*p\_df = map\_mem(id\_df); //fem el mapeig de la zona de memoria compartida

p\_df[0] = df[0]; //inicialitzem la zona de memoria compartida

p\_df[1] = df[1]; //inicialitzem la zona de memoria compartida

p\_df[2] = df[2]; //inicialitzem la zona de memoria compartida

p\_df[3] = df[3]; //inicialitzem la zona de memoria compartida

sprintf(str\_df,"%i",id\_df); //passem l'identificador de la zona de memoria compartida a un string

//fem el mateix per dc

char str\_dc[20];

int id\_dc = ini\_mem(sizeof(int)\*4); //creem la zona de memoria compartida

int \*p\_dc = map\_mem(id\_dc); //fem el mapeig de la zona de memoria compartida

p\_dc[0] = dc[0]; //inicialitzem la zona de memoria compartida

p\_dc[1] = dc[1]; //inicialitzem la zona de memoria compartida

p\_dc[2] = dc[2]; //inicialitzem la zona de memoria compartida

p\_dc[3] = dc[3]; //inicialitzem la zona de memoria compartida

sprintf(str\_dc,"%i",id\_dc); //passem l'identificador de la zona de memoria compartida a un string

//creem zona de memoria compartida per fantsmes

char str\_fantasmes[20];

int id\_fantasmes = ini\_mem(sizeof(objecte)\*n\_fantasmes); //creem la zona de memoria compartida

objecte \*p\_fantasmes = map\_mem(id\_fantasmes); //fem el mapeig de la zona de memoria compartida

for(i = 0; i < n\_fantasmes; i++){

p\_fantasmes[i] = fantasmes[i]; //inicialitzem la zona de memoria compartida

}

sprintf(str\_fantasmes,"%i",id\_fantasmes); //passem l'identificador de la zona de memoria compartida a un string

//fem el mateix amb mc

char str\_mc[20];

int id\_mc = ini\_mem(sizeof(objecte)); //creem la zona de memoria compartida

objecte \*p\_mc = map\_mem(id\_mc); //fem el mapeig de la zona de memoria compartida

\*p\_mc = mc; //inicialitzem la zona de memoria compartida

sprintf(str\_mc,"%i",id\_mc); //passem l'identificador de la zona de memoria compartida a un string

//fem el mateix d'abans amb la variable retard

char str\_retard[20];

int id\_retard = ini\_mem(sizeof(int)); //creem la zona de memoria compartida

int \*p\_retard = map\_mem(id\_retard); //fem el mapeig de la zona de memoria compartida

\*p\_retard = retard; //inicialitzem la zona de memoria compartida

sprintf(str\_retard,"%i",id\_retard); //passem l'identificador de la zona de memoria compartida a un string

//fem el mateix d'abans amb la variable retard

if(pthread\_create(&coco,NULL,mou\_menjacocos,NULL)!=0){

fprintf(stderr,"Error al crear el thread del menjacocos\n");

exit(1);

}

int n = 0;

char id\_proces[20];

//inicialitza\_joc();

fprintf(stderr,"\nhi han %d fantasmes\n",n\_fantasmes);

for(i = 0; i < n\_fantasmes; i++){

tpid[n] = fork();

if(tpid[n] == (pid\_t) 0){

sprintf(id\_proces,"%i",i); //passem l'identificador del thread a un string

if(execlp("./fantasmes3","fantasmes3",str\_fi1,str\_fi2,id\_proces,str\_fantasmes,str\_mc,str\_retard,str\_win,str\_n\_fil,str\_n\_col,(char \*)0)==-1){

fprintf(stderr,"Error al crear el proces del fantasma %d\n",i);

exit(0);

}

}else if(tpid[n] >0){

n++; //incrementem el numero de threads creats si és el pare

}else{

fprintf(stderr,"Error al crear el proces del fantasma %d\n",i);

exit(0);

}

}

//inicialitza\_joc();

/\*

\* FI CREACIÓ THREAD I PROCESSOS

\*/

do{

//actualitza\_pantalla();

win\_update();

win\_retard(100);

fi2 = \*p\_fi2;

\*p\_fi1 = fi1; //el controlem desde aquí

}while(!fi1&&!fi2);

pthread\_join(coco, (void \*\*)&fi1);

for(int th=0; th<n\_fantasmes;th++){

waitpid(tpid[th],&fi2,NULL); //esperem que el fill acabi

}

elim\_mem(id\_fi1); //eliminem la zona de memoria compartida

elim\_mem(id\_fi2); //eliminem la zona de memoria compartida

elim\_mem(id\_df); //eliminem la zona de memoria compartida

elim\_mem(id\_dc); //eliminem la zona de memoria compartida

elim\_mem(id\_win);

win\_fi();

if (fi1 == -1) printf("S'ha aturat el joc amb tecla RETURN!\n");

else { if (fi1) printf("Ha guanyat l'usuari!\n");

else printf("Ha guanyat l'ordinador!\n"); }

}

else

{

fprintf(stderr,"Error: no s'ha pogut crear el taulell:\n");

switch (rc)

{ case -1: fprintf(stderr,"camp de joc ja creat!\n");

break;

case -2: fprintf(stderr,"no s'ha pogut inicialitzar l'entorn de curses!\n");

break;

case -3: fprintf(stderr,"les mides del camp demanades son massa grans!\n");

break;

case -4: fprintf(stderr,"no s'ha pogut crear la finestra!\n");

break;

default: fprintf(stderr,"error desconegut, codi %d!\n",rc);

}

exit(6);

}

return(0);

}

*Fantasmes3.c*

/\*\*

\* @file fantasmes3.c

\* @brief Programa que implementa el comportament dels fantasmes, sent fils del programa principal

\* @author Assmaa Ouladali i David Domènech

\*

\*/

#include <stdio.h>

#include "memoria.h"

#include "winsuport2.h" /\* incloure definicions de funcions propies \*/

#include <stdlib.h> /\* per exit() \*/

typedef struct { /\* per un objecte (menjacocos o fantasma) \*/

int f; /\* posicio actual: fila \*/

int c; /\* posicio actual: columna \*/

int d; /\* direccio actual: [0..3] \*/

float r; /\* per indicar un retard relati \*/

char a; /\* caracter anterior en pos. actual \*/

} objecte;

//int \*df; /\* moviments de les 4 direccions possibles \*/

//int \*dc; /\* dalt, esquerra, baix, dreta \*/

int df[] = {-1, 0, 1, 0}; /\* moviments de les 4 direccions possibles \*/

int dc[] = {0, -1, 0, 1}; /\* dalt, esquerra, baix, dreta \*/

int \*fi1\_p, \*fi2\_p;

int fi1, fi2;

objecte \*mc; /\* informacio del menjacocos \*/

int \*retard;

int main(int n\_args, char \*ll\_args[]){

void \*p\_win;

int id\_fi1 = atoi(ll\_args[1]);

int id\_fi2 = atoi(ll\_args[2]);

//int id\_df = atoi(ll\_args[3]);

//int id\_dc = atoi(ll\_args[4]);

fi1\_p = map\_mem(id\_fi1);

fi2\_p = map\_mem(id\_fi2);

fprintf(stderr,"CAracteristiques exec fantasma:\t fi1: %d fi2: %d \n", fi1, fi2);

//df = map\_mem(id\_df);

//dc = map\_mem(id\_dc);

int i = atoi(ll\_args[3]);

int id\_f = atoi(ll\_args[4]); //posicio memoria compartida fantasmes

int id\_mc = atoi(ll\_args[5]); //la variable mc

int id\_retard = atoi(ll\_args[6]);

//int id\_win = atoi(ll\_args[9]);

//int fil = atoi(ll\_args[10]);

//int col = atoi(ll\_args[11]);

mc = map\_mem(id\_mc);

retard = map\_mem(id\_retard);

objecte \*fantasmes = map\_mem(id\_f); //punter a la memoria compartida fantasmes

objecte f1 = fantasmes[i];

fprintf(stderr,"CAracteristiques del fantasma:\t fila: %d col: %d dir: %d car: %c\n", f1.f, f1.c, f1.d, f1.a);

if(i < 0){

fprintf(stderr,"Error: index incorrecte\n");

exit(6);

}else{

fprintf(stderr,"Fantasma %d creat",i);

}

int id\_win = atoi(ll\_args[7]);

p\_win = map\_mem(id\_win);

if (p\_win == (int \*)-1)

{ fprintf(stderr,"proces (%d): error en identificador de finestra\n",(int)getpid());

exit(0);

}

int n\_fil = atoi(ll\_args[8]); /\* obtenir dimensions del camp de joc \*/

int n\_col = atoi(ll\_args[9]);

//win\_set((void \*)id\_win,n\_fil,n\_col); /\* crea acces a finestra oberta pel proces pare \*/

win\_set(p\_win,n\_fil,n\_col);

objecte seg;

int ret;

int k, vk, nd, vd[3];

ret = 0;

int z=0;

do{

z++;

fprintf(stderr,"CAracteristiques exec fantasma:\t fi1: %d fi2: %d \n", fi1, fi2);

//win\_update();

fprintf(stderr,"entra a fantasma per %d cop\n", z);

nd = 0; //numero de direccions disponibles

for (k=-1; k<=1; k++) /\* provar direccio actual i dir. veines \*/

{

vk = (f1.d + k) % 4; /\* direccio veina \*/

if (vk < 0) vk += 4; /\* corregeix negatius \*/

seg.f = f1.f + df[vk]; /\* calcular posicio en la nova dir.\*/

fprintf(stderr,"CAracteristiques exec fantasma:\t seg.f: %d f1.f: %d df[vk]: %d vk: %d\n", seg.f, f1.f, df[vk], vk);

seg.c = f1.c + dc[vk];

fprintf(stderr,"CAracteristiques exec fantasma:\t seg.c: %d f1.c: %d dc[vk]: %d vk: %d\n", seg.c, f1.c, dc[vk], vk);

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

fprintf(stderr,"CAracteristiques exec fantasma:\t seg.a: %c \n", seg.a);

if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0'))

{

vd[nd] = vk; /\* memoritza com a direccio possible \*/

nd++;

}

}

if (nd == 0){ /\* si no pot continuar, \*/

f1.d = (f1.d + 2) % 4; /\* canvia totalment de sentit \*/

}else

{

if (nd == 1){ /\* si nomes pot en una direccio \*/

f1.d = vd[0]; /\* li assigna aquesta \*/

}

else{ /\* altrament \*/

f1.d = vd[rand() % nd]; /\* segueix una dir. aleatoria \*/

}

seg.f = f1.f + df[f1.d]; /\* calcular seguent posicio final \*/

seg.c = f1.c + dc[f1.d];

seg.a = win\_quincar(seg.f,seg.c); /\* calcular caracter seguent posicio \*/

win\_escricar(f1.f,f1.c,f1.a,NO\_INV); /\* esborra posicio anterior \*/

fprintf(stderr,"CAracteristiques del fantasma:\t fila: %d col: %d dir: %d car: %c\n", f1.f, f1.c, f1.d, f1.a);

f1.f = seg.f;

f1.c = seg.c;

f1.a = seg.a; /\* actualitza posicio \*/

win\_escricar(f1.f,f1.c,'1'+i,NO\_INV); /\* redibuixa fantasma \*/

fprintf(stderr,"CAracteristiques del fantasma:\t fila: %d col: %d dir: %d car: %c\n", f1.f, f1.c, f1.d, f1.a);

if (f1.a == '0'){

ret = 1; /\* ha capturat menjacocos \*/

fprintf(stderr,"Fantasma %d capturat\n",i);

}

}

\*fi2\_p=ret;

fprintf(stderr,"CAracteristiques exec fantasma:\t fi1: %d fi2: %d \n", (\*fi1\_p), (\*fi2\_p));

win\_retard((mc->r)\* (int) \*retard);

}while (!(\*fi1\_p) && !(\*fi2\_p));//!fi1 && !fi2)

return (fi2);

}

### Joc de Proves:

Per a comprovar el correcte funcionament del nostre codi hem fet el següent joc de proves:

|  |  |  |
| --- | --- | --- |
| **Especificació:** | **Paràmetres** | **Resultat esperat** |
| Prova d’un tauler buit sense fantasmes, només amb el menjacocos. | * prova1.txt | El jugador pot moure el menjacocos pel mapa, no apareix cap fantasma.  El jugador guanya quan ha menjat tots els cocos. |
| Prova d’un tauler amb els següents paràmetres incorrectes:   * Direcció fantasma 1 fora de rang | * prova2.txt | El programa mostra un error i no es pot jugar. |
| Prova d’un tauler amb els següents paràmetres incorrectes:   * Fila fantasma 1 fora de rang | * prova3.txt | El programa mostra un error i no es pot jugar. |
| Prova de posar un fantasma sobre la paret. | * prova4.txt | El programa mostra un error i no es pot jugar. |
| Prova d’un tauler amb un nombre de fantasmes més gran que MAX\_FANTASMES. | * prova5.txt | No es permet jugar ja que hi ha una inconsistència en els paràmetres. |
| Prova d’un joc normal | * prova6.txt | Es permet jugar, quan el menjacocos és atrapat per un fantasma s’acaba la partida. |