Fonaments de Programació
PAC 3 - Tercera Prova d'Avaluació Continuada

Presentació

En aquest document es presenta la tercera prova d'avaluació continuada. Aquesta prova constitueix un 20% de la nota d'AC.

Competències

La principal competència del grau que es treballa en aquesta PAC és la capacitat de codificar i provar aplicacions informàtiques. Per treballar aquesta competència, la PAC planteja diversos exercicis al voltant de la traducció d'algorismes dissenyats en llenguatge algorísmic al llenguatge de programació C i l'habilitat de provar que aquests algoritmes funcionen.

Objectius

Els objectius concrets que es persegueixen en el desenvolupament de la PAC son els següents:

1. Codificar algorismes senzills en un llenguatge de programació, concretament en C.
2. Utilitzar eines d'edició i compilació. La proporcionada actualment per FP és el CodeLite
3. Depurar programes que no funcionen correctament.
4. Provar un programa – definir jocs de prova.
5. Passar correctament paràmetres d'entrada i sortida.
6. Utilitzar components existents.

Descripció de la PAC/pràctica a realitzar

La PAC consta de 3 exercicis, que treballen els diferents objectius que s'han enumerat com a claus en aquesta prova. Cada exercici consta d'un enunciat i un pes que indica el percentatge que representa respecte a la nota global.

Recursos

Serà necessari llegir prèviament el capítols del temari relatius a les nocions bàsiques d'algorísmica així com relatius als tipus bàsics de dades i el seu us, tant dels apunts de l'assignatura com de les TS que trobareu en les cites del calendari de la vostra aula.
Addicionalment, per resoldre els exercicis de codificació, caldrà que reviseu les taules de traducció i tot el material relacionat amb la codificació en C que trobareu a l'apartat de "recursos" de la vostra aula.

Criteris de valoració

S'avaluarà cada exercici d'acord amb el seu pes, que apareix indicat abans del seu enunciat. Els exercicis que constin de diversos apartats es valoraran com a completament correctes o incorrectes. Les respostes incorrectes no disminueixen la nota. Raoneu i justifiqueu totes les respostes, especialment si així ho indica l'enunciat (si, tot i demanar-ho l'enunciat, no justifiqueu la resposta, es comptabilitzarà com a incorrecte).

Format i data de lliurament

Cal lliurar la solució dels exercicis 1, 2 en un fitxer anomenat CognomsNom_FP_PAC3 adreçat a la bústia "Lliurament d'activitats" de l'aula de teoria. Cal lliurar la codificació de la pregunta 3 a través del Corrector Automàtic, que trobareu a l'aula de teoria.
Data límit per lliurar la solució: Dijous, 30 d'octubre de 2014 (a les 23:59 hores).

Exercici 1: Correcció d'errors de compilació i traducció [25%]

Objectius: Detectar i corregir errors de compilació, disseny i sintaxi.

Materials: M1 i M2 fins M2.1 i Manual de C

Tasca: L'algorisme següent llegeix un text de la seqüència d'entrada i en genera un altre resultat d'encriptar-lo amb una determinada clau. La clau d'encriptació a utilitzar es llegeix també de l'entrada estàndard (precedeix el text a encriptar). Així doncs, a l'entrada trobarem:

< Clau Text # >

On:

- *Clau*, un enter (entre 4 i 255) que s'utilitzarà per encriptar el text
- *Text*, una seqüència de caràcters (el text a encriptar).
- *#* és el caràcter que marca el final de la seqüència.

I, a la sortida, escriurem:

< TextEncriptat # >

On *TextEncriptat* és el text original al qual s'ha aplicat una transformació consistent a sumar la quantitat *Clau* al codi ASCII de cadascun dels caràcters.

```
const
        END:  caracter = '#';
fconst

algorisme  EncryptingAlgorithm
var
        key:  enter;
        car, encryptedCar: caracter;
fvar
        ReadTheKey( key, car );

        mentre no (car = END) fer
                encryptedCar:= EncryptChar( car, key );
                writeChar( encryptedCar );
                car:= readChar();
        fmentre

        writeChar( END ) ;
falgorisme
```

```
funcio EncryptChar( car: caracter, key: enter ):caracter
var
        code, encrypted: enter;
        encryptedCar: caracter;
fvar
        code:= charToCode( car );
        encrypted:= (256 + code + key) mod 256;
        encryptedCar:= codeToChar( encrypted );

        retorna encryptedChar;
ffuncio

accio ReadTheKey( sor key: enter, sor car: caracter )
        key:= readInteger();
        car:= readChar(); { saltem el caràcter separador }
        car:= readChar();
faccio
```

Us proporcionen la següent traducció a C té diversos errors sintàctics i de traducció.
NOTA: Per conveniència, els printf's de caràcters s'han traduït com a printf("%c", caracter), sense espai darrera el caràcter. Això és diferent a com s'indica a la taula de traducció. No ho considerarem un error a efectes d'aquest exercici.

Versió en llenguatge C amb errors:

```
/*-----------------------------------------------------------------
 *  Fonaments de Programacio - 2014-1
 *  Pac3 - Ex 1
 *-----------------------------------------------------------------*/

#include <stdio.h>

#define END #

typedef enum {FALSE, TRUE} bool;

int main(void)
{
     int key;
    char car, encryptedCar;

    ReadTheKey( &key, &car );

    while (car != END)
    {
        encryptedCar= EncryptChar( car, key );
        printf( "%c", encryptedCar );
        scanf("%c", &car);
    }
     return 0;
}
```

```
char EncryptChar( char car, int key )
{
    int code, encrypted;
    char encryptedCar;

    code= (int)(char);
    encrypted= (256 + code + key) % 256;
    encryptedCar= (char)( encrypted );

    return encryptedCar;
}

void ReadTheKey( int *key, char *car )
{
    scanf("%d", &key);
    scanf("%c", &car);
    scanf("%c", &car);
}
```

La vostra tasca consisteix en:

1. Identificar tots els errors de la traducció proporcionada i explicar-ne la causa.

2. Arreglar la traducció perquè funcioni correctament. Cal lliurar el codi complert i un resultat de la compilació.

3. Provar l'algorisme amb les següents dades, mostrant el resultat que es veu a la pantalla.

    a. 4 I have a dream#

    b. 255 YES,WE CAN#

    c. 32 ALERTA#

    d. 8 Con 10 cañones por banda#

    e. 4 0123456#

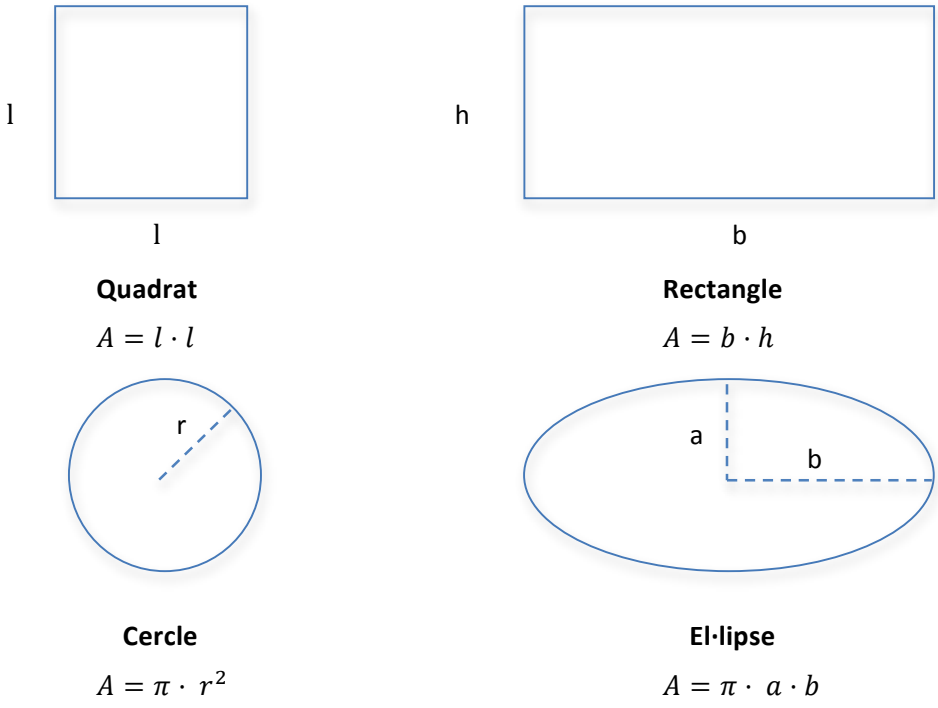
4. Noteu que el mateix algorisme d'encriptació podria servir per desencriptar missatges. Modifiqueu l'algorisme per tal que escrigui una sortida que pugui ser l'entrada pel mateix algorisme i que serveixi per recuperar el missatge original.

Exercici 2: Codificar un algorisme i saber-lo provar [45%]

Objectius: Mostrar l'habilitat en la traducció d'un algorisme de llenguatge algorísmic a C, i saber demostrar la seva correctesa.

Materials: M1 i M2 i Manual de C

Tasca: Disposem d'un algorisme que llegeix del canal estàndard una seqüència amb una sèrie de formes geomètriques, amb les seves dimensions, i en calcula l'àrea de cadascuna d'elles. Les figures suportades (i la manera de calcular la seva àrea) són:



Quadrat

$A = l \cdot l$

Rectangle

$A = b \cdot h$

Cercle

$A = \pi \cdot r^2$

El·lipse

$A = \pi \cdot a \cdot b$

La seqüència té el següent format:

< $Tipus_0$ $Param_0$ ... $Param_m$ $Tipus_1$ $Param_0$ ... $Param_p$ ... $Tipus_n$ Param ... $Param_q$ F >

On:

- *$Tipus_i$* és un caràcter que indica el tipus de la figura geomètrica *i*: S si és un quadrat, R si és un rectangle, C si és un cercle, i E si és una el·lipse.
- *$Param_j$* és un real que tindrà un significat diferent segons el tipus *i* de figura geomètrica. Si la figura *i* és S, hi haurà un sol paràmetre (la mida del costat, *l*). Si és R, hi haurà dos paràmetres (primer la base *b* i després l'alçada *h*). Si és C, hi haurà un paràmetre (el radi *r*). I, si és E, hi haurà dos paràmetres (semieix menor *a* i semieix major *b*).
- *F* és el caràcter que marca el final de la seqüència.

Per exemple, amb la seqüència d'entrada següent:

< S 5.0 R 5.0 2.0 C 2.5 E 3.0 6.0 F >

L'algorisme calcularà l'àrea d'un quadrat de 5 de costat, la d'un rectangle de 5 de base i 2 d'alçada, la d'un cercle de 2.5 de radi i, finalment, la d'una el·lipse de semieixos 3 i 6. La sortida, tenint en compte que els reals es generen amb un decimal, serà:

< 25.0 10.0 19.6 56.5 F >

L'algorisme, a més, comprova la coherència dels paràmetres que defineixen les diferents formes geomètriques de manera que, si es detecta qualsevol error en ells, l'àrea a retornar serà 0. L'algorisme és:

```
const
        SQUARE: caracter = 'S';
        RECTANGLE: caracter = 'R';
        CIRCLE: caracter = 'C';
        ELLIPSE: caracter = 'E';
        PI: real = 3.141592;
        END: caracter = 'F';
fconst

algorisme areaCalculator
var
        type, blank: caracter;
        l, r, a, b, h: real;
        area: real;
fvar
        area := 0.0;
        type:= readCharacter();

        mentre no (type = END) fer
                si type = SQUARE llavors
                        l:= readReal();
                        area:= CalculateAreaForSquare( l );
                sino    si type = RECTANGLE llavors
                                b:= readReal();
                                h:= readReal();
                                area:= CalculateAreaForRectangle( b, h );
                        sino    si type = CIRCLE llavors
                                        r:= readReal();
                                        area:= CalculateAreaForCircle( r );
                                sino    si type =  ELLIPSE llavors
                                                a:= readReal();
                                                b:= readReal();
                                                area:= CalculateAreaForEllipse( a, b );
                                        fsi
                                fsi
```

```
                    fsi
                fsi

                writeReal( area ); { Traduïu com a printf("%.1f ", area); }
                blank:= readCharacter();
                type:= readCharacter();
        fmentre

        writeChar( END );

falgorisme

funcio CalculateAreaForSquare( l: real ): real
var
        area: real;
fvar
        si (l > 0.0) llavors
                area:= l*l;
        sino
                area:= 0.0;
        fsi
        retorna area;
ffuncio

funcio CalculateAreaForRectangle( b: real, h: real ): real
var
        area: real;
fvar
        si (b > 0.0 i h > 0.0) llavors
                area:= b*h;
        sino
                area:= 0.0;
        fsi
        retorna area;
ffuncio

funcio CalculateAreaForCircle( r: real ): real
var
        area: real;
fvar
        si (r > 0.0) llavors
                area:= PI * r*r;
        sino
                area:= 0.0;
        fsi
        retorna area;
ffuncio
```

```
funcio CalculateAreaForEllipse( a: real, b: real ): real
var
        area: real;
fvar
        si (a > 0.0 i b > 0.0 i a < b) llavors
                area:= PI*a*b;
        sino
                area:= 0.0;
        fsi
        retorna area;
ffuncio
```

Tasques:

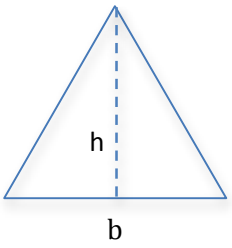
1. Codifiqueu l'algorisme en C. Annexeu el codi C a la solució i una còpia de pantalla del resultat de la compilació.

2. Quins jocs de prova dissenyaríeu per comprovar que l'algorisme funciona correctament en tots els casos?

3. Si volguéssim prescindir de la funció CalculateAreaForSquare, podríem utilitzar la funció CalculateAreaForRectangle per calcular l'àrea dels quadrats? En cas afirmatiu, indiqueu com ho faríeu. Contesteu en llenguatge C.

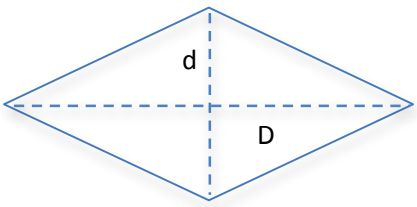
4. Si volguéssim prescindir de la funció CalculateAreaForCircle, podríem utilitzar la funció CalculateAreaForEllipse per calcular l'àrea dels cercles? En cas afirmatiu, indiqueu com ho faríeu. Contesteu en llenguatge C.

5. Indiqueu quines modificacions caldria incorporar en l'algorisme per tal de suportar dos nous tipus de geometria: el triangle (tipus 'T') i el rombe (tipus 'B'). El triangle té dos paràmetres: base (b) i alçada (h). El rombe en té també dos: la diagonal major (D) i la diagonal menor (d). Contesteu en llenguatge C.



Triangle

$$A = \frac{b \cdot h}{2}$$



Rombe

$$A = \frac{D \cdot d}{2}$$

Exercici 3: Codificar un algorisme amb paràmetres de sortida i crides a accions predefinides [30%]

Objectius: Mostrar l'habilitat en la traducció d'un algorisme que inclou paràmetres de sortida, de llenguatge algorísmic a C.

Materials: M1 i M2 i Manual de C

Tasca: Codificar el següent algorisme en C.

Una gran superfície comercial disposa d'un algorisme per calcular l'import de les compres dels seus clients. L'algorisme llegeix una seqüència on hi apareixen tots els articles de la compra, així com els descomptes i promocions aplicables. La seqüència d'entrada té el següent format:

< TC
$codi_0$ $preu_0$ qtt_0 $oferta_0$ $tipus_0$ $codi_1$ $preu_1$ qtt_1 $oferta_1$ $tipus_1$...$codi_n$ $preu_n$ qtt_n $oferta_n$ $tipus_n$ -1 >

On:

- *TC* és un caràcter que diu si el client pagarà amb targeta client ('S') o no ('N'). Pagar amb targeta client suposa un descompte directe d'un 5% sobre l'import total de la compra.
- $codi_i$ és un enter positiu que representa el producte *i*.
- $preu_i$ és un real positiu que indica el preu unitari del producte *i.*
- qtt_i és un enter positiu que indica el nombre d'unitats del producte *i.*
- $oferta_i$ és un caràcter que indica si hi ha alguna promoció vigent sobre el producte *i*. Si és 'N', no hi ha cap oferta. Si és 'T' hi ha una promoció del tipus "3 x 2", és a dir, paguem 2 unitats i ens en duem 3. Si és 'D' hi ha una promoció del tipus "2 x 1", és a dir, paguem 1 unitat i ens en duem 2. Per últim, si és 'M' hi ha una promoció del tipus "Segona unitat a meitat de preu".
- $tipus_i$ és el tipus d'IVA a aplicar sobre el producte *i* (enter entre 0 i 100).

El següent algorisme llegeix la seqüència i treu pel canal estàndard l'import total de la compra:

```
const
        ENDSEQ:  enter = -1;
        PROMO_NONE:  caracter = 'N';
        PROMO_3x2 :  caracter  = 'T';
        PROMO_2x1:  caracter = 'D';
        PROMO_HALF: caracter = 'M';
        PROMO_LOYALTY: real = 0.95;
fconst
```

```
algorisme  computeTotalPrice
var
        loyaltyCard:  boolea;
        code, qtt, percent:  enter;
        promo, card, blank:  caracter;
        price, VAT, amount:  real;
fvar
        amount:= 0.0;
        card:= readCharacter();
        loyaltyCard:=  (card = 'S');
        blank:= readCharacter();
        code:=readInteger();
        mentre  code ≠ ENDSEQ  fer
                price:= readReal();
                qtt:= readInteger();
                blank:= readCharacter();
                promo:= readCharacter();
                blank:= readCharacter();
                percent:= readInteger();
                VAT:= 1.0 + (integerToReal( percent ) / 100.0);
                addToAmount( amount, price, qtt, promo, VAT );
                code:=readInteger();
        fmentre
        si  loyaltyCard  llavors
            amount:= amount * PROMO_LOYALTY;
        fsi
        writeReal(amount); { Traduïu com a printf("%.2f ",amount); }
falgorisme

accio  addToAmount(   entsor amount:  real, ent  price: real,
                      ent qtt:  enter, ent promo: caracter, ent VAT: real )
var
        productPrice :  real ;
fvar
        productPrice:= 0.0;
        si promo = PROMO_NONE llavors
                productPrice:= computeNoPromoPrice( price, qtt, VAT );
        sino    si promo = PROMO_3x2 llavors
                        productPrice:= computeNxMPromoPrice( 3, 2, price, qtt, VAT );
                sino    si promo =  PROMO_2x1 llavors
                                productPrice:= computeNxMPromoPrice( 2, 1, price, qtt, VAT );
                        sino
                                productPrice:= computeHalfPromoPrice( price, qtt, VAT );
                        fsi
                fsi
        fsi
        amount:= amount + productPrice;
faccio
```

```
funcio computeNoPromoPrice( price: real,  qtt:  enter, VAT: real ): real
        retorna price * integerToReal( qtt ) * VAT;
ffuncio

funcio computeNxMPromoPrice( N: enter, M: enter, price: real,  qtt:  enter, VAT: real ): real
var
        promo, noPromo: real;
fvar
        promo:= integerToReal(qtt div N) * integerToReal(M) * price * VAT;
        noPromo:= integerToReal(qtt mod N) * price * VAT;
        retorna promo + noPromo;
ffuncio

funcio computeHalfPromoPrice( price: real,  qtt:  enter, VAT: real ): real
var
        promo, noPromo: real;
fvar
        noPromo:= (integerToReal( qtt div 2 ) + integerToReal( qtt mod 2 )) * price * VAT;
        promo:= integerToReal( qtt div 2 ) * price * 0.5 * VAT;
        retorna promo + noPromo;
ffuncio
```

Tasca: La vostra tasca és codificar l'algorisme en C i lliurar-lo al corrector automàtic