Introducció a la programació

M. Jesús Marco Galindo
Josep Vilaplana Pastó

PID_00149890

Universitat Oberta de Catalunya
www.uoc.edu

Índex

Introducció

Aquest mòdul introdueix el món de la programació com a disciplina de l'enginyeria. A partir d'aquí es comprendran millor els objectius que cal assolir per a esdevenir un bon programador. I, de fet, això és, com veureu, l'objectiu d'aquesta assignatura.

La finalitat bàsica de la programació és solucionar problemes mitjançant l'ordinador. Com en qualsevol disciplina, l'aprenentatge ha de ser progressiu: primer cal aprendre a resoldre problemes senzills a partir d'un conjunt d'eines bàsiques, per a més endavant resoldre problemes més complexos que requereixen més eines.

Així, doncs, aquest material està format per un seguit de mòduls amb una estructura orientada a l'aprenentatge progressiu dels continguts i, per tant, és molt important assimilar els continguts d'un mòdul abans de passar al següent.

Abans d'entrar en matèria cal saber de què parlem. Per a poder seguir el discurs de l'assignatura és indispensable conèixer, en primer lloc, el significat dels conceptes amb els quals treballarem. En el primer apartat d'aquest mòdul definirem aquests conceptes bàsics.

A partir dels conceptes podrem fer un pas endavant i passar al segon apartat, que ens presenta la programació com una disciplina de l'enginyeria i ens mostra de quina manera cal emprendre el desenvolupament d'un programa per a esdevenir un bon professional. Penseu que la programació no és un art, sinó una tècnica. Amb tot això, ja serem capaços d'entendre els objectius de l'assignatura, que s'exposen en el tercer apartat.

Un cop situats i coneguts tots aquests aspectes, en el mòdul "Introducció a l'algorísmica" estarem en condicions de començar l'aprenentatge de les eines i mètodes que ens calen per a desenvolupar un programa.

Objectius

Els objectius d'aquest mòdul són bàsicament els següents:

1. Comprendre els conceptes bàsics de programació, algorisme i programa.

2. Conèixer les etapes bàsiques de desenvolupament d'un programa.

3. Entendre la diferència entre disseny i implementació i comprendre la importància que el disseny d'algorismes té en la programació.

4. A partir de la comprensió dels conceptes bàsics, conèixer quins són els objectius de l'assignatura.

1. Conceptes bàsics de programació

Per a entendre en què consisteix la programació com a disciplina i quins són els objectius d'aquesta assignatura ens cal abans de res comprendre els conceptes bàsics que haurem d'utilitzar. Així, doncs, els definirem d'una manera senzilla i clara.

Vegeu els objectius dels fonaments de la programació a l'apartat 3 d'aquest mòdul.

1.1. Definicions

Situem-nos primer en un dels conceptes més importants amb els quals treballarem: el concepte d'*algorisme*.

> Un **algorisme** es defineix com una descripció no ambigua i precisa de les accions que cal fer per a resoldre un problema ben definit en un temps finit.

Definició d'*algorisme* segons...

... el *Diccionari de la Llengua* de la Gran Enciclopèdia Catalana:

"Un algorisme és un procediment de càlcul que consisteix a acomplir un seguit ordenat i finit d'instruccions que condueix, un cop especificades les dades, a la solució que el problema genèric en qüestió té per a les dades considerades".

Per tant, podem pensar en un algorisme com una recepta o guió que cal seguir per a resoldre un problema determinat, normalment a partir d'una informació que tenim d'entrada (per exemple, una recepta de cuina). Un algorisme, però, és un mètode general per a resoldre tots els casos possibles del mateix problema i, per tant, ha de ser independent de les dades d'entrada de qualsevol cas concret. 

Per a comprendre completament aquest concepte d'*algorisme* ens caldrà definir ara els conceptes d'*entorn*, *acció*, *procés* i *processador*.

L'**entorn** és el conjunt d'objectes necessaris per a portar a terme una tasca determinada. L'**estat de l'entorn** en un moment determinat és la descripció de l'estat dels objectes de l'entorn en aquell moment concret. Un algorisme actua de manera que fa canviar progressivament l'estat del seu entorn.

Exemple d'entorn i estat

En el cas d'una recepta de cuina, l'entorn vindria donat pels estris de cuina (olles, paelles, etc.) i els ingredients.

L'estat de les paelles, per exemple, canviarà de netes a brutes.

Una **acció** és un esdeveniment finit en el temps i que té un efecte definit i previst. Una acció pot actuar sobre un entorn i el pot modificar, és a dir, es parteix d'un estat inicial i s'arriba a un estat final diferent. Una **acció elemental** és una acció que el destinatari d'un algorisme entén i sap processar.

Un **procés** és l'execució d'una o diverses accions. L'algorisme expressa unes pautes que cal seguir per a portar a terme una tasca concreta. L'encarregat de dur a terme aquesta tasca és el processador. Un **processador** és una entitat capaç de comprendre i executar eficaçment un algorisme. El destinatari de l'algorisme és, doncs, el processador.

Un processador pot ser una persona, una rentadora, un ordinador, etc.

Ara que ja sabem què és un algorisme ens cal decidir com el podem expressar. Haurem de trobar un llenguatge que ens permeti fer una descripció no ambigua i precisa de les accions que componen els nostres algorismes.

Anomenem **llenguatge natural** el llenguatge que normalment utilitzem per a comunicar-nos. Ara bé, atesa la seva complexitat i ambigüitat, veurem que el llenguatge natural no ens permet definir les accions amb la precisió i claredat que volem. De fet, si nosaltres actuem com a processadors d'algú que ens indica com s'ha de fer una tasca, sovint demanem puntualitzacions i aclariments de què cal fer.

El català, el castellà o l'anglès serien exemples de llenguatges naturals.

Necessitem, doncs, un llenguatge més reduït i precís. En els apartats següents descriurem un llenguatge concret creat expressament per a poder expressar qualsevol algorisme amb la claredat i precisió necessàries. L'anomenarem **llenguatge algorísmic** o **notació algorísmica**.

Llenguatges no naturals

No és la primera vegada que us trobeu en la situació d'haver d'utilitzar una notació nova (diferent del llenguatge natural) per a expressar conceptes; per exemple, esteu molt acostumats a emprar la notació matemàtica (+, –, >, π, log, etc.).

En aquesta assignatura ens dedicarem a aprendre les tècniques bàsiques per a dissenyar algorismes. Per tant, ens caldrà aprendre aquest llenguatge algorísmic que acabem d'esmentar. En el mòdul "Introducció a l'algorísmica", descrivim el llenguatge algorísmic i, en la resta l'aprenem a fer servir de manera progressiva conjuntament amb les tècniques per al disseny d'algorismes.

Arribats en aquest punt ja hem definit la majoria dels conceptes principals amb els quals treballarem; i fixeu-vos que, curiosament, encara no hem parlat de programes.

Definirem un programa en relació amb tot allò que hem definit fins ara. Per a fer-ho, ens caldrà aclarir també què és un ordinador o computador.

Encara que ja tenim una idea del que és, un **ordinador** es pot definir formalment com una màquina formada per circuits electrònics que té la capacitat de resoldre problemes sota el control d'unes instruccions donades. Un ordinador està format bàsicament per un **processador**, la **memòria** i els **dispositius d'entrada i sortida** que permeten la seva comunicació amb l'exterior.

Exemples de llenguatges de programació...

... són Pascal, C, C++, Cobol, Fortran, VisualBasic, Java, etc.

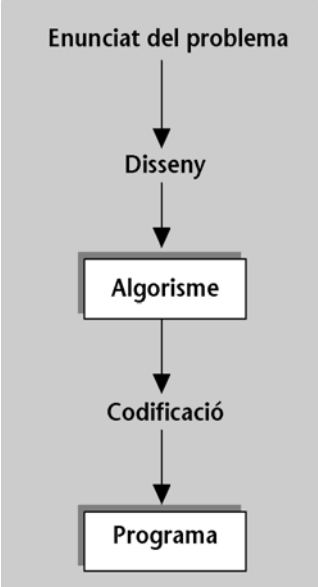
L'ordinador processarà els nostres algorismes, però per a fer-ho ha d'entendre el nostre llenguatge algorísmic. Ens caldrà, doncs, transcriure els nostres algorismes a un **llenguatge de programació**, és a dir, a un llenguatge capaç de ser comprès per un ordinador.

Definició d'ordinador

En aquest punt, podem definir també un **ordinador** com un autòmat de càlcul governat per un programa.

Així, doncs, un **programa** és només la codificació d'un algorisme en un llenguatge que l'ordinador entén.

Per a expressar i dissenyar algorismes emprem sempre la notació algorísmica; d'altra banda, l'ús i definició de llenguatges de programació estan subjectes a factors de disponibilitat, tecnologia actual, etc.



En aquesta assignatura, a mesura que aprenem a dissenyar algorismes amb el nostre llenguatge algorísmic, aprendrem també a implementar-los en un llenguatge de programació perquè es puguin executar en el nostre ordinador.

Els materials corresponents a la codificació en un llenguatge de programació els trobareu a l'aula. Amb l'estudi d'aquests materials podreu implementar programes. Us aconsellem que comenceu a partir dels dissenys que trobareu en aquests mòduls.

Bàsicament, hi ha dos tipus de llenguatges de programació: els **llenguatges imperatius** o **llenguatges procedimentals** i els **llenguatges declaratius**, que a la vegada es divideixen en **llenguatges funcionals** i **llenguatges lògics**.

Exemples de llenguatges de programació

a. Exemples de llenguatges imperatius: Pascal, Cobol i C.

b. Exemples de llenguatges declaratius:

- Llenguatges funcionals: Lisp.
- Llenguatges lògics: Prolog.

En la programació imperativa, els programes són seqüències d'instruccions que s'han de dur a terme com una recepta o guió per a resoldre un problema determinat.

En aquesta assignatura estudiarem només la programació imperativa, que tradicionalment ha estat la més estesa i utilitzada.

Alguns llenguatges de programació faciliten el seguiment d'una metodologia concreta de programació. Una de les més àmpliament acceptada actualment és l'orientació a l'objecte*.

*** Per exemple, C++ i Object Pascal són, respectivament extensions de C i Pascal que faciliten l'ús d'aquesta metodologia.**

L'**orientació a l'objecte** determina un estil de programació que es caracteritza per la manera de manipular la informació.

En assignatures posteriors veureu el paradigma de l'orientació a l'objecte aplicat a la programació. En aquesta assignatura introduïm els fonaments bàsics que anireu aprofundint més endavant.

En l'assignatura *Programació orientada a l'objecte* s'explica en què consisteix la programació sota el paradigma de l'orientació a l'objecte.

Fixeu-vos, però, que el realment important és arribar a saber dissenyar un algorisme que resolgui un problema determinat. El fet de codificar-lo per a obtenir-ne el programa corresponent consisteix simplement a fer una traducció.

El llenguatge algorísmic que definirem i utilitzarem per a dissenyar algorismes és prou genèric perquè resulti força senzill fer la codificació en qualsevol llenguatge de programació imperatiu.

A mesura que avanceu en aquesta matèria entendreu més a fons algunes d'aquestes definicions que pot ser que inicialment no us quedin del tot clares. Convé, doncs, que contínuament repasseu aquest mòdul.

1.2. Exemples

En aquest subapartat veurem alguns exemples que ens permetran aclarir de manera intuïtiva els conceptes que acabem de definir. De fet, alguns dels conceptes que hem vist són aplicables a qualsevol entorn del món real, i no estan restringits a l'entorn de la informàtica.

Podríem pensar en una rentadora com un autòmat, ja que és una màquina capaç de fer una feina de manera autònoma, en aquest cas, rentar la roba.

Si demanem a la nostra rentadora que ens renti la roba blanca, la rentadora seguirà el procés següent:

- Agafar aigua i sabó del calaix.
- Escalfar l'aigua a 40 graus.
- Donar voltes durant 20 minuts (rentar).
- Expulsar l'aigua.
- Agafar més aigua.
- Donar voltes durant 10 minuts (primera esbandida).
- Expulsar l'aigua.

  …
- Donar voltes durant 10 minuts (quarta esbandida).
- Expulsar l'aigua.
- Donar voltes molt ràpidament durant 1 minut (centrifugar).

Tot aquest conjunt d'accions que la rentadora ha dut a terme seria l'algorisme que aquest aparell segueix per a rentar la roba blanca.

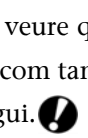
Si ara demanem que ens renti la roba delicada, les accions que executarà la rentadora seran segurament diferents (per exemple, la temperatura serà inferior, el temps de rentatge serà més curt, hi haurà menys esbandides, etc.); això equival a dir que l'algorisme per a rentar la roba delicada és diferent de l'algorisme per a rentar la roba blanca.

Altres exemples d'algorismes

De la mateixa manera que en el cas esmentat de la rentadora, també podríem pensar en un algorisme que resolgui el problema de fer una truita de patates, el de canviar la roda d'un cotxe, el de construir un edifici, el de preparar la declaració de la renda, el de multiplicar dues matrius, etc.

Així, doncs, un algorisme ens indica les accions que cal seguir per a resoldre un problema concret. Però ens cal un autòmat (processador) que sigui capaç d'executar-lo, i un entorn afectat (roba, aigua, bombo, etc.).

També és important veure que hi pot haver més d'un algorisme que resolgui el mateix problema, com també algun problema per al qual no hi hagi cap algorisme que el resolgui.



És important que recordeu això quan parlem de l'especificació, en el mòdul "Introducció a l'algorísmica" d'aquesta assignatura.

Fixeu-vos també que cal que es compleixin unes condicions inicials perquè el problema es resolgui correctament (que la roba blanca estigui dins la rentadora, que la rentadora estigui connectada al corrent elèctric i a l'entrada d'aigua, i també que hi hagi sabó al calaixet). Per tal de poder arribar a l'estat final desitjat, que en el nostre cas seria tenir la roba neta al final, l'entorn hauria d'estar en aquest estat inicial. Si no partim de l'estat inicial correcte, segurament no aconseguirem el nostre objectiu (penseu, per exemple, que si la rentadora no està connectada al corrent, el nostre algorisme no donarà el resultat esperat).

Tornant ara al nostre entorn, l'ordinador és l'autòmat que ha de ser capaç de resoldre els problemes que li plantegem*.

*** Per exemple, que ens calculi la mitjana de quatre nombres enters determinats.**

Un algorisme en llenguatge natural per a resoldre el problema podria ser el següent: "Sumeu els quatre números que ens han donat i dividiu el resultat d'aquesta suma per quatre per a obtenir-ne el resultat final".

Ara veurem, a tall d'exemple, com s'expressaria aquest algorisme en llenguatge algorísmic; més endavant, quan coneguem el llenguatge algorísmic, ja l'entendrem millor. Posteriorment, podríem implementar aquest algorisme en un llenguatge de programació i executar-lo en el nostre ordinador.

```
algorisme mitjana
    var
        n, suma, i: enter;
        resultat: real;
    fvar
    suma := 0;
    i := 1;
    mentre i ≤ 4 fer
        llegirEnter(n);
        suma := suma + n;
        i := i + 1;
    fmentre
    resultat := enterAReal(suma)/4.0;
    escriureReal(resultat);
falgorisme
```

2. La programació com a disciplina d'enginyeria

Tot el que estudiarem en aquesta assignatura ens ha de servir per aprendre a programar correctament. Vegem, doncs, com a motivació per a l'estudi de la programació, què es fa en el món real.

Actualment, la **programació** es considera una disciplina que, igual que altres disciplines d'enginyeria, es fonamenta en una teoria, una proposta metodològica i un conjunt de tècniques de disseny.

Cadascuna d'aquestes parts de la disciplina ha sorgit de la recerca i l'experiència adquirides en els darrers anys i contribueix a fer que la programació sigui una tasca eficaç (doni els resultats esperats) i eficient (en un temps de desenvolupament raonable).

Evidentment, en un entorn industrial competitiu, l'**eficàcia** i l'**eficiència** són valors molt apreciats en les empreses. Precisament per aquest motiu l'assignatura s'orienta a aprendre a programar de manera eficient i a adquirir un cert grau de destresa a l'hora de construir programes correctes en un temps raonable. No obstant això, per a aconseguir-ho, us caldrà practicar molt.

Per a comprendre més bé els objectius concrets de l'assignatura, veurem ara a grans trets quines són les etapes que clàssicament se segueixen per a desenvolupar un programa.

Vegeu els objectius que s'especifiquen a l'apartat 3 d'aquest mòdul didàctic.

2.1. Etapes en el desenvolupament d'un programa

Per a obtenir un programa que resolgui un problema cal passar per les etapes següents:

1. Definició del problema. Anàlisi de requeriments
2. Disseny de l'algorisme
3. Implementació del programa
4. Proves
5. Operació, millores i manteniment

Observació

Recordeu que ens movem dins el paradigma de la programació imperativa. Si seguíssim d'altres tipus de programació, el cicle de desenvolupament podria tenir diferències.

Després de l'etapa de proves, ja tenim un programa disponible per a començar a operar-hi. Però com a resultat de la seva explotació i manteniment, és possible que calgui fer-hi millores o canvis, en definitiva, modificacions que generalment provoquen haver de retrocedir a algunes etapes prèvies del desenvolupament.

Per motius d'eficiència, és desitjable que el desenvolupament de cadascuna de les etapes s'efectuï de manera seqüencial en el temps i que cap etapa obligui a retrocedir a l'anterior (excepte, com ja hem dit, la darrera). Avui dia es disposa de coneixements suficients perquè això es pugui portar a terme.

En els subapartats següents descriurem cadascuna de les etapes.

2.1.1. Definició del problema. Anàlisi de requeriments

Un o diversos clients plantegen un problema que necessita una solució mitjançant la construcció d'un programa. D'entrada, cal fer una anàlisi del problema, que pot ser laboriosa, ja que a vegades el problema és difícil de comprendre. Un cop s'ha aclarit el problema, es fa una anàlisi minuciosa dels requeriments que ajudaran a definir el problema que volem resoldre.

El resultat de l'**etapa de definició del problema i anàlisi de requeriments** serà un enunciat precís i clar del problema que cal resoldre.

En aquesta assignatura no estudiarem aquesta etapa (no es pot aprendre tot de cop!), ja en tindreu oportunitat en assignatures posteriors. Per tant, partirem sempre d'un enunciat ja complet, clar i concret, i ens centrarem en les tres etapes que descriurem a continuació.

2.1.2. Disseny de l'algorisme

A partir de l'enunciat hem de fer un disseny que ens porti a la solució desitjada.

Progressivament, al llarg de cada mòdul aprendreu les tècniques, eines i metodologies adients per a poder fer aquesta etapa amb eficàcia i eficiència. La representació del disseny es farà mitjançant el llenguatge algorísmic que s'exposa bàsicament en el mòdul "Introducció a l'algorísmica" d'aquesta assignatura.

Com veurem, aquesta és una de les etapes més importants i costoses dins el desenvolupament d'un programa. A més, els objectius d'aquesta assignatura se centren principalment en l'aprenentatge d'aquesta etapa. Per tant, en el darrer apartat d'aquest mòdul veurem amb més detall en què consisteix el disseny d'un algorisme.

2.1.3. Implementació del programa

Implementar vol dir *portar a terme*. El resultat del disseny és un algorisme, però per a executar-lo, caldrà traduir el disseny fet en llenguatge algorísmic a un llenguatge de programació que l'ordinador entengui. Aquesta etapa és senzilla, atès que es tracta només d'una traducció força mecànica del disseny ja fet per a obtenir el programa corresponent, que podrem executar a l'ordinador.

Exemple d'implementació del llenguatge

Si escollim implementar els nostres algorismes en C, només ens caldrà conèixer la sintaxi i la semàntica del llenguatge C, i la manera com es tradueix cada operació del llenguatge algorísmic a la sentència corresponent en C.

Podrem traduir sense cap dificultat a llenguatges de programació imperatius qualsevol disseny que hàgim fet en llenguatge algorísmic. Simplement haurem d'establir les regles de traducció al llenguatge escollit i procedir-hi.

Aprendreu l'etapa d'implementació en el llenguatge de programació escollit per a codificar els algorismes a la part pràctica de l'assignatura. El material d'aquesta part pràctica no està inclòs en aquests mòduls, sinó que el trobareu a l'aula corresponent.

2.1.4. Proves

Finalment, caldrà detectar possibles errades en els nostres programes.

En l'etapa de proves es tracta de provar el programa resultant amb diferents dades d'entrada que reben el nom de **jocs de proves**. L'èxit d'aquestes proves dependrà en gran mesura de la qualitat del disseny fet abans.

Tingueu en compte, però, que els jocs de proves només serveixen per a comprovar que el programa no és correcte (si algun joc de proves no dóna els resultat esperat), però no per a assegurar que sí que ho és (llevat que es comprovin totes les entrades possibles, que és pràcticament impossible).

De fet, encara que en aquesta assignatura no els estudiarem, hi ha mètodes formals de verificació que permeten demostrar la correctesa d'un algorisme –i per tant, del programa corresponent– a partir de l'especificació formal dels seus requeriments. D'aquesta manera es podria afirmar amb tota seguretat que el programa compleix els seus objectius.

Correctesa del programa

La millor manera d'assegurar que el programa és correcte és estar-ne convençuts de la correctesa en l'etapa de disseny. La metodologia i les pautes de raonament d'algorismes que es proposen en els mòduls següents d'aquesta assignatura us ajudaran a assolir la correctesa del disseny.

De totes maneres, a falta de la verificació formal d'algorismes, en aquesta assignatura es donen tècniques i pautes metodològiques que poden portar al "convenciment" que s'ha fet un disseny correcte, i l'elaboració d'un joc de proves bo i exhaustiu*, confirmarà les nostres pretensions amb una seguretat alta. Fixeu-vos, però, que la seguretat en la correctesa ens ve donada per les dues etapes: la primera (disseny) és la més estratègica, i la segona (implementació i prova amb un joc de proves exhaustiu), la de confirmació.

*** Per exemple, que verifiqui els casos estranys o extrems.**

2.1.5. Operació, millores i manteniment

Podeu pensar que un cop obtingut i provat el programa, el procés ja s'ha acabat. Però de la mateixa manera que encara es dissenyen cotxes nous, rentadores noves, etc., els programes també canvien per a millorar o adaptar-se a les noves necessitats, i per aquesta raó es torna a reproduir tot el procés des de l'inici.

Si la nova millora obliga a modificar parts del programa o si, per qüestions de manteniment, s'hi han de fer correccions, això només es pot dur a terme amb eficiència si el disseny del programa es fa de manera intel·ligible.

Un programa s'escriu una vegada, però es llegeix moltes vegades més. Convé, doncs, de fer dissenys clars, comprensibles i senzills perquè ajudin a fer que aquesta etapa sigui eficient i eficaç.

El disseny també haurà d'estar ben documentat. Per tant, caldrà aprendre a documentar bé els programes i saber distingir entre una documentació bona i una documentació que no aporta res.

2.2. Conclusions i motivació

Les etapes anteriors tenen costos molt diferents:

1) L'**anàlisi de requeriments** és una de les més costoses, ja que depèn de molts factors entre els quals, a més dels purament tècnics, es troben els sociològics i econòmics.

2) L'**etapa de disseny** és costosa i cal dedicar-hi tots els esforços que calgui, atès que de la qualitat del disseny dependrà el cost i l'èxit de la resta d'etapes.

3) L'**etapa d'implementació** és actualment la menys costosa de tot el desenvolupament, atès que, com ja hem dit, és un traducció força directa.

Conèixer perfectament un llenguatge de programació no vol dir saber programar. Conèixer un llenguatge de programació és conèixer l'eina a partir de la qual podrem implementar el disseny del programa. Així, doncs, hem d'esforçar-nos a saber dissenyar bé els algorismes; la resta serà una tasca molt menys costosa.

4) El **cost de l'etapa de proves** depèn de la complexitat que tingui el programa i del fet que les etapes anteriors s'hagin fet bé.

En tot cas, recordeu que en aquesta etapa no es pot estar mai segur del funcionament correcte i que fer un bon disseny ajudarà a tenir la seguretat que el programa implementat funciona correctament. Potser pensareu que això és una mica exagerat, però no és així. Tingueu present que els errors d'un programa en funcionament poden aturar la producció d'una gran empresa, amb els greus perjudicis econòmics que això pot ocasionar.

La importància de la correctesa és clara

En algunes aplicacions com l'hospitalària o la informàtica de control d'una central nuclear, els errors poden tenir efectes dramàtics.

Podríem pensar que, quan ja sabem com es desenvolupa un programa en una empresa i un cop fets alguns programes, ja sabem programar. Però amb això no n'hi ha prou. Per a ser bons programadors, capaços d'afrontar problemes de gran envergadura com els que ens trobarem al món professional, caldrà adquirir molta destresa en l'aplicació dels mètodes i les tècniques que exposarem. Aquesta assignatura i les que la segueixen, combinades amb la pràctica contínua, us permetran assolir aquesta destresa per a ser bons professionals.

Algú podria dir que...

... sap jugar a tennis perquè ha llançat una pilota amb una raqueta. Però per a arribar a ser un professional d'alt nivell, s'ha de tenir destresa en un conjunt de tècniques i mètodes que cal dominar per a ser competitiu. El mateix passa en el món de la programació.

3. Objectius de l'assignatura

En aquesta assignatura ens centrem bàsicament en el disseny d'algorismes i la seva posterior implementació a un llenguatge de programació imperatiu per a obtenir el programa corresponent.

Com ja sabem...

... l'etapa de disseny és una de les més costoses en el desenvolupament d'un programa.

A partir d'un enunciat concret haurem d'aprendre a fer el disseny d'un algorisme. D'altra banda, també hem vist que hi pot haver més d'un algorisme que solucioni un mateix problema. Per tant, hem de tenir uns criteris clars per a escollir l'algorisme més adient en cada cas.

Un algorisme adient serà aquell que presenti les característiques següents:

a) Correctesa: l'algorisme fa allò que realment es demana.

b) Intel·ligibilitat: l'algorisme ha de ser clar i fàcil d'entendre, atès que s'escriurà un sol cop, però caldrà llegir-lo molts més per a poder-lo mantenir i/o modificar.

c) Eficiència: l'algorisme ha de portar a terme la tasca que se li ha encomanat en un temps raonable.

d) Generalitat: amb pocs canvis, l'algorisme s'ha de poder adaptar a altres enunciats semblants.

Quan dissenyeu els vostres algorismes tingueu en compte sempre aquests quatre criteris. La metodologia que proposarem en aquesta assignatura us ajudarà a dissenyar algorismes que compleixin aquests criteris. Però penseu també que no només és necessari seguir la metodologia, sinó que per a assolir una certa destresa en el disseny cal força pràctica, ja que tant els conceptes com la metodologia necessiten un temps d'assimilació, i això només s'aconsegueix amb la pràctica contínua.

3.1. Etapes del disseny d'un algorisme

Les etapes per a dissenyar un algorisme són pràcticament les mateixes que calen per a resoldre qualsevol problema d'altres disciplines. Bàsicament són les següents:

1) Entendre el problema: si no entenem el problema, difícilment el podrem resoldre. L'especificació formal dels algorismes és una bona eina per a assolir aquesta etapa rigorosament i evitar ambigüitats.

Ara bé, en aquesta assignatura farem servir llenguatge natural per a especificar els nostres algorismes. Per tant, serà una especificació informal i haurem de tenir cura d'expressar-la sense ambigüitat i amb rigor. En l'algorisme que en resultarà, l'especificació apareixerà en forma de comentaris.

L'especificació d'algorismes es descriu en el mòdul "Introducció a l'algorísmica" d'aquesta assignatura.

2) Plantejar/planificar la solució: la metodologia proposada (bàsicament els esquemes de recorregut i cerca i l'anàlisi descendent) ens permetrà efectuar de manera eficient i amb eficàcia aquesta etapa i alhora facilitarà la resta d'etapes.

Els esquemes de recorregut i cerca s'estudien en el mòdul "Tractament seqüencial" i la tècnica de disseny descendent en el mòdul "Introducció a la metodologia de disseny descendent" d'aquesta assignatura.

3) Formular la solució: la notació o llenguatge algorísmic ens permetrà definir l'algorisme amb precisió i sense ambigüitats.

El llenguatge algorísmic es descriu en el mòdul "Introducció a l'algorísmica" d'aquesta assignatura.

4) Avaluar la correcció de la solució proposada: la intel·ligibilitat de l'algorisme junt amb l'aplicació dels esquemes ens permetrà estar gairebé segurs de la correcció de l'algorisme. L'especificació (precondicions, postcondicions, etc.) de l'algorisme també ens ajudarà, a més, a raonar sobre la correctesa de l'algorisme.

3.1.1. Entendre el problema

Cal assegurar-se que hem llegit correctament l'enunciat. Cal saber què es demana i de quin estat inicial es parteix. En aquesta etapa, es pretén ordenar les idees, identificar quines són les condicions inicials, que anomenarem **precondició**, i a partir d'aquí, establir l'objectiu que es vol assolir, que anomenarem **postcondició**.

L'especificació d'algorismes s'estudia en el mòdul "Introducció a l'algorísmica" d'aquesta assignatura.

Quan definim els objectius i les condicions inicials, no podem permetre cap ambigüitat, encara que utilitzem llenguatge natural. L'**especificació** consistirà precisament en això.

Un **algorisme** és una descripció clara i precisa d'accions que cal fer per a resoldre un problema ben definit en un temps finit. L'algorisme ha de solucionar estrictament el que demana l'enunciat. Ni més ni menys. Així, doncs, en aquesta etapa cal fixar les idees que ens permetran de continuar en les etapes successives.

No ens hem de preocupar de com ho farem, sinó de què hem de fer, què tenim inicialment i què volem obtenir.

Exemple d'especificació del problema

L'objectiu d'una rentadora és rentar la roba que hi hem introduït. Per a assolir aquest objectiu cal que estigui connectada al corrent i al subministrament d'aigua. Tot i que aquest exemple és força simple, fixeu-vos que a l'hora de fer aquesta especificació de la rentadora, en cap moment no hem pensat com s'ho fa per a netejar la roba, sinó només en què ha de fer.

En el mòdul "Introducció a l'algorísmica" aprofundirem aquest concepte.

3.1.2. Plantejar i planificar la solució

Aquesta és l'etapa de disseny a la qual cal dedicar més temps.

El primer que s'ha de fer és veure si el problema correspon a un tipus de problema que ja s'ha estudiat en la teoria. Si és així, s'apliquen els esquemes coneguts per a arribar a resoldre'l d'una manera segura i ràpida.

En el mòdul "Tractament seqüencial" d'aquesta assignatura s'estudien diferents esquemes de resolució per a alguns tipus de problemes.

Tingueu en compte que si no apliqueu els esquemes teòrics de resolució, trigareu més a trobar la solució, ja que haureu de redescobrir per vosaltres mateixos la teoria i, a més, no estareu del tot segurs de la correctesa de la solució que proposeu.

Per a resoldre problemes complexos, no en tindrem prou amb l'aplicació d'esquemes; per a això ens caldrà aplicar la metodologia de disseny descendent. Aquesta metodologia ens permetrà abordar problemes de més complexitat i convertirà el problema complex en un conjunt de problemes més petits als quals podrem aplicar tot el que hàgim après per a problemes de complexitat menor.

Vegeu la tècnica de disseny descendent en el mòdul "Introducció a la metodologia de disseny descendent".

3.1.3. Formular la solució

La notació algorísmica ens permet expressar de manera precisa i clara l'algorisme que proposem.

L'objectiu bàsic del mòdul "Introducció a l'algorísmica" és precisament exposar la sintaxi i semàntica del llenguatge algorísmic que presentem al llarg de l'assignatura.

Per a fer-ho, cal conèixer perfectament la sintaxi i la semàntica dels elements del llenguatge algorísmic proposat. Sense aquest coneixement, difícilment podríem proposar o expressar amb precisió l'algorisme definitiu.

La formulació de l'algorisme és el resultat d'aplicar correctament la metodologia suggerida, no pas la combinació més o menys capritxosa dels elements i/o construccions del llenguatge algorísmic. És a dir, en l'etapa anterior hem pensat el disseny més enllà dels elements del llenguatge algorísmic.

3.1.4. Avaluar la correcció de la solució

Fer el seguiment d'un algorisme per a un cas concret de dades d'entrada no ens permet assegurar que l'algorisme funciona per a tots els casos.

La metodologia proposada en els mòduls d'aquesta assignatura, juntament amb l'especificació informal que afegirem com a comentari en les parts de l'algorisme dissenyat, ens permetrà raonar sobre la seva correctesa.

3.2. Implementació d'un programa

Un cop tinguem el disseny de l'algorisme, el codificarem per a obtenir un programa. La codificació s'ha d'introduir a l'ordinador en el llenguatge de programació escollit mitjançant un editor que crearà un fitxer de text anomenat **fitxer font**.

Aquesta etapa s'aprèn amb les pràctiques del laboratori i amb l'ajuda del material complementari d'aquests mòduls. Aquest material us ensenyarà a traduir les sentències del llenguatge algorísmic al llenguatge de programació escollit i a emprar el programari necessari (editor i compilador) per a poder compilar i executar els vostres programes.

El text que hem introduït en aquest fitxer font no és directament executable per l'ordinador; per a poder-lo executar, l'hem de convertir en instruccions que el processador pugui entendre directament. L'encarregat de fer aquesta tasca és un programa anomenat **compilador** que traduirà el codi font a un codi que l'ordinador comprèn.

Posteriorment, un altre programa s'encarregarà d'efectuar el muntatge i d'obtenir un fitxer que es pot executar i provar. Un dels objectius del **programa muntador** és fer l'adaptació del codi generat a l'entorn de l'ordinador perquè es pugui executar.

Resumint, les **etapes d'implementació d'un programa** són les següents: traduir o codificar l'algorisme al llenguatge de programació, editar el programa, compilar-lo, muntar-lo i executar-lo.

En l'etapa de compilació poden sorgir errors de sintaxi del llenguatge de programació i caldrà tornar a editar el programa per a corregir-los. Un cop superats els errors de compilació, el muntatge no hauria de donar cap problema.

Per a l'etapa d'execució, és molt positiu dissenyar un bon joc de proves que inclogui totes les possibilitats a les quals pensem que el programa ha de fer front.

Quan es produeixen errors en l'execució del programa, segurament voldrà dir que no s'ha fet correctament alguna de les etapes prèvies.

Aparició d'errors en l'execució del programa

En l'execució del programa poden aparèixer errors perquè no ens hem fixat bé en la traducció o en l'edició del programa, o perquè que hi hagi hagut alguna equivocació a l'hora de planificar i plantejar el disseny de l'algorisme. En el darrer cas, haurem d'estudiar molt bé el disseny d'algorismes i posar més cura en les primeres etapes de disseny. Recordeu que l'esforç realitzat i el temps dedicat en aquestes primeres etapes fa que la resta d'etapes siguin molt més senzilles i directes.

Podríem caure en la temptació de fer les etapes a l'inrevés o bé de deixar-nos-en alguna. És a dir, editar un programa directament, compilar-lo, executar-lo i, si funciona, fer-ne el disseny. Aquest sistema, conegut com *assaig i error*, crea programes enrevessats a causa de les modificacions introduïdes en la idea original durant cada fracàs del programa. Com a conseqüència, el programa és poc o gens intel·ligible. Recordeu que la programació és una disciplina de l'enginyeria i, per tant, cal seguir una metodologia per a no caure en el parany de l'assaig i error, i així aprendre a programar amb el rigor necessari.

Us acabem de presentar els objectius que cal assolir en aquesta assignatura. Seria bo que els rellegiu de tant en tant durant el curs per tal de no perdre'ls mai de vista. Més endavant també els podreu entendre millor.

Resum

Amb els conceptes que hem introduït fins ara, ja estem en condicions d'enumerar l'objectiu general d'aquesta assignatura: aprendre a dissenyar algorismes per a resoldre problemes de complexitat mitjana i a implementar el programa corresponent dins el paradigma de la programació estructurada i imperativa.

Podríem desglossar aquest objectiu en els següents: 

1) Aprendre a entendre i especificar, encara que informalment, els enunciats dels algorismes.

2) Aprendre un llenguatge algorísmic com a eina per a expressar algorismes.

3) Aprendre diferents tècniques de disseny, com ara els esquemes, per a resoldre problemes senzills, i la metodologia de disseny descendent per a solucionar problemes de més envergadura.

4) Aprendre a fer servir les estructures de dades adients a cada problema.

5) Aprendre a combinar les eines i tècniques exposades per a resoldre problemes.

6) Aprendre a implementar els algorismes en un llenguatge de programació procedimental per a obtenir un programa.

7) Aprendre a utilitzar el programari necessari (editor i compilador) per tal de poder editar, compilar i executar els programes.

8) Aprendre a dissenyar jocs de proves que permetin de validar la correctesa dels programes.

Els primers cinc objectius s'assoleixen de manera gradual a partir de l'estudi dels mòduls didàctics combinats amb la realització dels exercicis i activitats proposats. Els tres darrers, a partir de fer pràctiques i l'ús del laboratori, del programari i del material complementari de l'aula.

Glossari

algorisme
Conjunt explícit de regles per a resoldre un problema en un nombre finit de passos.

codificació
Traducció d'un algorisme a un llenguatge de programació.

computador
Autòmat de càlcul governat per un programa.

entorn
Conjunt d'objectes necessaris per a dur a terme una tasca determinada.

especificació
Formalització de l'enunciat d'un problema.

estat
Descripció de l'entorn en un moment determinat.

implementació
Realització d'un projecte.

llenguatge algorísmic
Llenguatge artificial que es fa servir per a expressar algorismes.

llenguatge natural
Qualsevol llenguatge que es fa servir com a forma natural de comunicació: català, castellà, anglès, etc.

llenguatge de programació
Llenguatge creat per a expressar programes i que és capaç de ser interpretat per un ordinador.

notació algorísmica
Vegeu *llenguatge algorísmic*.

ordinador
Vegeu *computador*.

programa
Codificació d'un algorisme en un llenguatge de programació que l'ordinador entengui.

sintaxi
Regles d'estructura d'un llenguatge.

semàntica
Significat dels símbols d'un llenguatge.

Bibliografia

Botella, P.; Bofill, M.; Burgués, X.; Franch, X.; Lagonigro, R; Vancells, J. (1998). *Fonaments de programació I*. Barcelona: Ediuoc.

Castro, J.; Cucker, F.; Messeguer, X.; Rubio, A.; Solano, L.; Vallés, B. (1992). *Curs de programació*. Madrid, etc.: McGraw-Hill.

Vancells, J.; López E. (1992). *Programació: introducció a l'algorísmica*. Vic: Eumo Editorial (Tecno-Ciència).