Pascal

Autor: Albert Teira Osuna
Assignatura: Llenguatges de programació
Quadrimestre de primavera del curs 2018/19

Creació i desenvolupament

Història

Va ser desenvolupat pel professor suís Niklaus Wirth (Institut tecnològic de Zurich, Suïssa) al 1971. El nom és en honor a Blaise Pascal, matemàtic, físic i filòsof francès que va contribuir a disenyar les calculadores mecàniques, va fer nombroses aportacions a la teoria de la probabilitat, entre moltes altres coses.
Prèviament a treballar en Pascal, Niklaus Wirth va desenvolupar Euler i ALGOL W, i, a posteriori va desenvolupar Modula-2 i Oberon.
Al començament, Pascal va néixer com una idea per millorar l'ensenyança de tècniques de programació a l'alumnat de la universitat on era professor. Però, es va començar a extendre i pràcticament es va convertir en el llenguatge de programació de tota una generació d'estudiants.

Pascal a les aules

Tal i com he mencionat abans, Pascal va ser un llenguatge utilitzat en gran quantitat a les aules universitàries amb un únic i gran propòsit: millorar les maneres de programar dels alumnes.
I per què? Doncs ben senzill, per les següents tres raons:
- Llenguatge molt estructurat
- Sintaxi senzilla
- Fa una comprovació exhaustiva de tipus

És un llenguatge força llegible i entendible de manera que és molt útil per introduïr a algú a la programació i també per a aquells que ja en coneixen les bases però necessiten o volen millorar el seu estil de programar.

Versions preliminars i primers compiladors

La primera versió de Pascal va aparéixer al 1968, però no va ser fins al 1971 quan es va publicar de forma oficial el llenguatge amb el seu primer compilador. El problema va venir quan les diferents versions que es llençaven no eren compatibles al 100% les unes amb les altres, això va fer que es posés en marxa un projecte d'estandardització del llenguatge que va acabar amb dos en comptes d'un projecte:
- El primer, l'ISO Pascal, de l'International Standard Organization (ISO) al 1982.
- El segon, l'ANSI/IEEE Pascal, un projecte conjunt entre l'American National Standards Institute (ANSI) i l'Institute of Electrical and Electronics Engineers (IEEE).

La diferència entre ells no és gaire significativa, però qui realment va guanyar-se el mercat va ser Turbo Pascal, un compilador i entorn de treball integrat (IDE) per Pascal, desenvolupat per Borland i llençat al 1983. Inicialment el van treure per MS-DOS, CP/M, CP/M-86 i més endavant per Windows.

Turbo Pascal

Amb l'aparició de Turbo Pascal i el gran boom que va tenir, es van continuar treient versions fins al 1992. Al llarg d'aquests 9 anys es van arribar a treure fins a 7 versions, cadascuna amb una sèrie de millores respecte l'anterior:
- Turbo Pascal 1.0, 1983.
- Turbo Pascal 2.0, 1984.
- Turbo Pascal 3.0, 1985.
- Turbo Pascal 4.0, 1987.
- Turbo Pascal 5.0, 1988.
- Turbo Pascal 5.5, 1989.
- Turbo Pascal 6.0, 1990.
- Turbo Pascal 7.0, 1992.

Afegir que aquest éxit, en una part important, va ser degut al Apple Pascal. Ja que la major part dels programes desenvolupats per al Mac estaven en Pascal i també una bona part dels programes de MS-DOS. A més, el preu de Turbo Pascal estava en 49 dòlars mentre que el compilador de Pascal de Microsoft valia més de 100.

Un cop arribat a mitjans dels 90, amb l'aparició de Java, l'estandardització del llenguatge C i el constant creixement d'altre llenguatge com el C++, Pascal va passar a un segon pla. Un altre fort cop per al Pascal va ser que Apple va baixar el suport també.
Amb Turbo Pascal a les últimes, i, en conseqüència Pascal també, es va dur a terme un projecte que intentava fer un mix entre Pascal i Object Pascal, i el resultat va ser Delphi.

Delphi

És un entorn de desenvolupament de software que utilitza com a llenguatge de programació l'Object Pascal. És multiplataforma, soporta Windows, macOS, iOS, Android i Linux. El seu principal ús és el desenvolupament d'aplicacions visuals i de bases de dades client-servidor, però es pot utilitzar per a desenvolupar qualsevol cosa pràcticament, com per exemple apps per consola o apps de web. Una de les aplicacions més populars que han estat creades amb Delphi i Object Pascal és Skype.
Object Pascal va ser una evolució de Pascal en la qual s'introduïa el paradigma orientat a objectes entre d'altres coses. Va ser desenvolupat per Apple amb el suport de Niklaus Wirth, creador de Pascal.

Paradigmes de programació

Què són els paradigmes de programació?

Buscant per internet he trobat diferents definicions, però potser la que més m'ha agradat ha estat la següent:

> Un paradigma de programació indica un mètode de realitzar còmputs i la manera en que s'han d'estructurar i organitzar les tasques que ha de dur a terme un programa.

Pel que sé i he vist, els llenguatges de programació solen utilitzar no un, si no diversos d'ells. I, pel general, cada paradigma s'associa a un estil de programar. A continuació faig una menció als paradigmes més comuns que podem trobar:
- Paradigma imperatiu: descriu *com* s'ha de realitzar el càlcul.
- Paradigma declaratiu: descriu *què* s'ha de calcular.
- Paradigma funcional
- Paradigma lògic
- Paradigma orientat a objectes

Paradigmes de programació que fa servir Pascal

Pascal bàsicament fa ús del paradigma **imperatiu**. Dins d'aquest paradigma trobem altres paradigmes dels quals Pascals fa ús de:
- Paradigma procedural
- Paradigma estructurat

Mencionar que en el cas d'Object Pascal també s'utilitza el paradigma de la programació orientada a objectes. A continuació entrarem en més detall amb aquests tres paradigmes de programació.

Paradigma imperatiu

És un dels paradigmes més representatius. Va ser el primer paradigma acceptat. Tal i com el seu nom indica, imperatiu significa donar ordres, i això és el que fan aquells llenguatges que són imperatius.
Les principals característiques que tenen els llenguatges imperatius són:
- Expressen **com** s'ha de realitzar el càlcul.
- Un programa està format per un conjunt d'instruccions que s'executen seguint un flux de control. Aquestes instruccions **modifiquen** l'estat del programa.
- Tenim **variables**, les quals s'emmagatzemen a memòria i poden ser modificades durant l'execució d'un programa.
- Aquestes variables contenen dades o referències que estableixen l'estat del programa. Per tant, el resultat d'un programa correspon amb l'estat final que tenen les variables.
- La instrucció principal és l'**assignació**.
- Podem executar un conjunt d'instruccions de manera repetida.
- Basat en el model de còmput de màquines de Turing.

Paradigma procedural

Aquest paradigma es basa principalment en aglutinar tots els conjunts d'instruccions repetits en un o més d'un programa en un procediment o funció. D'aquesta manera, cada cop que volguem executar aquella part del codi simplement hem de cridar al procediment o funció que toqui.
Fent ús de funcions ens estalviem moltes repeticions i fem els programes molt més curts i comprensibles. En termes de la màquina no aconseguim cap millora, és únicament un paradigma que ens manté l'estructura d'un programa molt més neta i ordenada.

Paradigma estructurat

Aquest paradigma es troba orientat a millorar el desenvolupament d'un programa. Es fa ús de:
- Blocs d'instruccions
- Condicionals (*if* i *switch*)
- Iteració (*for* i *while*)

S'aconsegueixen grans avantatges, com per exemple:
- Facilita el treball simultani entre diferents programadors
- Reutilització de codi

Compilat o interpretat

Pascal és un llenguatge **compilat**. Què vol dir que sigui un llenguatge compilat? Doncs que el codi font, que ha estat escrit en un llenguatge d'alt nivell (en aquest cas Pascal), passa per un compilador que el tradueix a un llenguatge que la màquina entèn. Aquesta traducció queda guardada en un arxiu que serà el que executarem.
Els compiladors de Pascal més destacables, ja sigui actualment o perquè ho van ser en el seu moment, són:
- Delphi
- Free Pascal
- GNU Pascal
- Turbo Pascal
- Oxygene

L'avantatge dels llenguatges compilats és que un cop hem compilat el nostre codi font, ja no cal tornar-ho a fer i, per tant, podem executar el programa tantes vegades com vulguem reduïnt el temps d'espera entre diverses execucions.
A diferència dels llenguatges interpretats, que en comptes de tenir un compilador, tenen un intèrpret que fa la traducció d'alt nivell a llenguatge màquina a cada execució, instrucció per instrucció. Òbviament això comporta una espera major entre execucions, però també té avantatges

com per exemple: ens permet un tipat dinàmic de dades i es té una gran independència respecte la plataforma on s'executa.

Sistema de tipus

Segons la definició donada a classe, un sistema de tipus és un conjunt de regles que assignen *tipus* als elements d'un programa per evitar errors. Podem clasificar un llenguatge segons si és *type safe* o *type unsafe*. Que sigui *type safe* vol dir que cap programa donarà errors de tipus en temps d'execució. En el cas de Pascal trobem que és *type unsafe* ja que si fem servir un apuntador alliberat, ens pot donar errors de tipus.
Una altra característica del sistema de tipus d'un llenguatge és si té un *tipat fort* o un *tipat feble*. Amb un *tipat fort* s'evita barrejar valors de diferents tipus, d'aquesta manera ens evitem molts problemes. És a dir, no podem fer servir el valor d'una variable d'un tipus com si fos d'un altre tipus diferent a no ser que fem un *typecast*. Tampoc podem aplicar una operació a un valor que no sigui d'un tipus que permeti aplicar-la. En aquest cas, Pascal el té *fort*.
Com a tercera característica trobem de nou una altra classificació. En aquest cas entre *manifest typing* i *inferred typing*. Pascal és *manifest*, això vol dir que s'ha d'especificar explícitament el tipus de cadascuna de les variables declarades. Per contra, en un llenguatge amb *inferència* de tipus, s'auto detecta el tipus de les dades.
Finalment, la última característica a determinar per al sistema de tipus és si el llenguatge fa servir una comprovació de tipus *estàtica* o *dinàmica*. L'*estàt ica* es porta a terme en temps de compilació, en canvi, la *dinàmica* ho fa en temps d'execució. Doncs Pascal té una comprovació de tipus *estàtica*. Això comporta que en Pascal, cada variable i cada paràmetre té un tipus associat.

Principals aplicacions

Tal i com ja hem comentat amb anterioritat en els primers apartats, Pascal va néixer com un llenguatge dirigit a alumnes i no com un llenguatge enfocat al desenvolupament professional. L'objectiu que tenia Niklaus Wirth quan va crear-lo, era el de facilitar l'aprenentatge de programar als seus propis alumnes, fent ús de la programació estructurada que té Pascal. Però, l'ús d'aquest llenguatge va traspassar la frontera acadèmica i mica en mica va aconseguir convertir-se en una eina per crear i desenvolupar

qualsevol tipus de programa. Aquesta gran popularitat va arribar gràcies a les seves característiques que fan que sigui un llenguatge fàcilment llegible i molt estructurat, tal i com hem comentat abans.

Llenguatges semblants

Té forces aspectes en comú amb C++. Tots dos són *type unsafe*, tenen un *tipat fort* i tenen una comprovació de tipus *estàtica*. La principal diferència és que C++ soporta la programació orientada a objectes, cosa que Pascal no.
N'hi ha de molts altres, però penso que cadascún d'ells té el seu estil i no hi ha una forma exacta de declarar que siguin "semblants".

Exemples de codi

A continuació, exposaré una sèrie de blocs de codi que ajudaràn a comprendre millor tota la sintaxi de Pascal i veure com podem fer un programa.
Abans de començar, mencionar les tres parts principals que té un programa en Pascal:
- Program: paraula reservada que explicita el començament d'un programa. No és estrictament necessari el seu ús.
- Var: part on es declaren les variables. Pots declarar diferents variables d'un mateix tipus separant-les per comes.
- Begin - End: paraules reservades que marquen l'inici i el final d'un bloc d'instruccions a executar.

A més, podem trobar altres paraules reservades que ens permeten declarar funcions, procedures, etc.

Input / Output

En el següent exemple podem veure les tres partes ben diferenciades. A *va r* tenim les variables *num1* i *num2*, que són enters. A continuació, trobem el bloc de les instruccions a executar. En aquest cas ens trobem amb:
- write / writeln: ens permet escriure per pantalla.
- readln: ens permet llegir per pantalla.

```
program input_output;
var
```

```
    (* declaració de variables *)
    num1, num2 : integer;

begin
    write('Input the first number:');
    readln(num1);
    writeln('Input the second number:');
    readln(num2);
end.
```

Funcions

En aquest cas, veiem com es declara una funció en Pascal. Primerament, tenim les paraules reservades *program* i *var* que fan referència al programa principal i tal i com hem vist a l'exemple anterior, ens permet declarar variables. La diferència la trobem ara, en comptes de tenir el bloc *begin - end* del programa principal, trobem la paraula *function*. Aquesta etiqueta és la que ens permet la declaració de la funció. Té la següent estructura:

```
function name(argument(s): type1; argument(s): type2; ...): func
var declarations;

begin
    ...
    < statements >
    ...
    name:= expression;
end;
```

La variable que ens permet emmagatzemar el que torna la funció es diu igual que el nom de la funció. A continuació he fet un petit exemple real d'un programa amb una funció.

```
program exemple_funcio;
var
    a, b, ret : integer;

(*definició de la funció que ens torna el mínim entre dos nombre
function min(num1, num2: integer): integer;
var
    (* declaració de variables locals de la funció *)
    result: integer;

begin
```

```
    if (num1 < num2) then
        result := num1
    else
        result := num2;

    min := result;
end;

begin
    a := 5;
    b := 10;

    (* crida a la funció *)
    ret := min(a, b);

    writeln('El mínim és: ', ret);
end.
```

Procedures

Ara explicarem els procedures. La diferència entre un procedure i una funció és que en aquest cas, no es retorna res. Però es poden passar variables per referència i modificar-les en el propi procedure. L'estructura que té és la següent (il·lustrada en un exemple real ja):

```
program exemple_procedure;
var
    a, b, ret : integer;

(*definició del procedure que troba el mínim entre dos nombres*)
procedure min(num1, num2: integer; var resultat: integer);

begin
    if (num1 < num2) then
        resultat := num1
    else
        resultat := num2;
end;

begin
    a := 5;
    b := 10;

    (* crida al procedure *)
    min(a, b, ret);
```

```
    writeln('El mínim és: ', ret);
end.
```

Veiem com en aquest cas, en comptes de retornar el valor del mínim, tenim una variable que la passem per referència i fiquem el valor en aquesta mateixa. Per passar una variable per referència l'únic que hem de fer és posar la paraula reservada *var* davant de la variable que volguem passar per referència a la capçalera del procedure / funció.

Recursivitat

També podem utilitzar recursivitat en Pascal, quan una funció es crida a sí mateixa. En aquest cas he fet el càlcul del factorial per a un nombre entrat per l'usuari per pantalla.

```
program exemple_recursivitat
var
    n, resultat: integer;

function factorial(x: integer): integer;

begin
    if x = 0 then
        factorial := 1
    else
        factorial := x * factorial(x-1);
end;

begin
    writeln('Entra un número per calcular el seu factorial:');
    readln(n);
    resultat := factorial(n);
    writeln('El factorial de ', n, ' és ', resultat);
end.
```

Bucles

Òbviament també tenim bucles, els podem fer servir amb la construcció *while .. do*. Aquesta construcció avalua la condició abans d'executar l'interior, per tant, si de primeres la condició ja no es compleix, mai s'arribarà a executar el bucle. El que hem fet per poder escriure un bloc d'instruccions dins del bucle és anidar un *begin .. end* a dins. L'exemple mostrat a continuació mostra com fer un petit bucle.

```
program exemple_bucle
var
    n: integer;

begin
    writeln('Entra un número per escriure tots els valors des de
    readln(n);
    while n <> 0 do
    begin
        writeln(n)
        n := n - 1
    end;
end.
```

Altres característiques del llenguatge

Tipus

Una característica clara de Pascal és el fixat del tipus d'una variable. Això és fa en el moment de definir la variable i d'aquesta manera esquivem forces errors.
Si ens fixem en els tipus de les variables trobem que en Pascal hi han 7 tipus de variables bàsiques:
- Caràcter
- Enter
- Real
- Booleà
- Enumeració
- Subrang
- String

Llibreries

Les llibreries en Pascal són tal i com les coneixem actualment, una agrupació de procediments i funcions segons el context d'aquests. Per fer ús de llibreries en un programa hem d'utilitzar la paraula reservada *uses*. A continuació un petit exemple:

```
uses CRT;

var
```

```
    num: integer;

begin
    clrscr;
    num := 5;
    writeln(num);
end.
```

En aquest petit programa fem ús de la llibreria *CRT*. Una de les múltiples funcions de les que disposem en aquesta llibreria és *clrscr* que neteja la pantalla de la terminal. Per tant, aquest programa, esborra la pantalla de la terminal i a continuació escriu el número 5.

Fonts d'informació

Gran part de la informació extreta per a l'elaboració d'aquest docuement s'ha obtingut d'internet. Totes les pàgines consultades estan citades a continuació a la bibliografia. La majoria de pàgines, però, són o bé blogs informatius o bé transparències d'alguna universitat. També comentar que he seguit un petit tutorial de Pascal (també està a la bibliografia) que m'ha servit per veure el funcionament d'aquest llenguatge i conèixer-lo en més profunditat. Els exemples de codi els he creat jo basant-me en alguns trobats a internet.

Bibliografia

Omar Lara, Kevin. (2014). Lenguajes de programación: Paradigma imperativo. *Portafolio digital de Kevin Omar Lara.* Recuperat de https://kevinldp.wordpress.com/paradigma-imperativo/

Pascal Tutorial. (Sense data). Recuperat de https://www.tutorialspoint.com/pascal/index.htm

Soep666. (2011). Historia del lenguaje Pascal: Breve historia y definiciones. *Historia Pascal Richard.* Recuperat de http://historiapascalrichard.blogspot.com/

Vaca Rodríguez, César. (2012-2013). Universidad de Valladolid: *Paradigm as de programación.* Recuperat de https://www.infor.uva.es/~cvaca/asigs/docpar/intro.pdf

Wikibooks. Programación en Pascal / Historia. (Actualitzat per darrer cop al 2013). Recuperat de https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Pascal/Historia

Wikipedia, la enciclopedia libre. Pascal (lenguaje de programación). (Actualitzat per darrer cop al 2019). Recuperat de https://es.wikipedia.org/wiki/Pascal_(lenguaje_de_programaci%C3%B3n)