STUDIA UNIVERSITATIS
Revistă științifică a Universității de Stat din Moldova, 2011, nr.9(49)

TENDINȚE DE DEZVOLTARE A LIMBAJELOR DE PROGRAMARE ORIENTATE PE OBIECTE DIN PERSPECTIVA VIITORULUI PROFESOR DE INFORMATICĂ

Silviu GÎNCU

Universitatea de Stat din Tiraspol

In this article we provide a brief analysis of modern programming languages used in the educational process of pedagogical universities. We analyze this current trend in computer science as an object-oriented programming, as well as offerring a comparative analysis of language in this direction.

În evoluția sa, omenirea a cunoscut mai multe salturi esențiale în dezvoltare. O etapă esențială a fost apariția calculatorului. Se părea că odată cu apariția acestuia toate problemele vor fi soluționate. Însă, s-a dovedit că calculatorul este doar o maşină care primeşte comenzi şi le execută. Aşadar, odată cu apariția calculatoarelor apare necesitatea de a găsi un mijloc de comunicare între om şi sistemul de calcul. Acest mijloc a fost programul. Un program este o succesiune de instrucțiuni ce vor fi executate de sistemul de calcul. Pe baza programelor au fost elaborate alte programe, care, la rândul lor, „creau programe". De aici apare şi noțiunea de „produs program". Prin produs program se înțelege o multitudine de instrucțiuni prin intermediul cărora utilizatorul „comunică" cu sistemul de calcul. Este absolut necesar ca un produs program să fie eficient, pentru ca „comunicarea" dintre utilizator şi calculator să decurgă fără anomalii. Un produs program va fi reuşit dacă:

- satisface sau chiar depăşeşte aşteptările utilizatorului;
- a fost dezvoltat în limite de timp şi de buget rezonabile;
- este flexibil la modificări şi uşor de adaptat.

Un produs program este creat prin intermediul altor programe. În dependență de programul prin intermediul căruia este creat produsul, acesta va corespunde sau nu cerințelor inițiale. La elaborarea acestora sunt utilizate mai multe tehnologii, una dintre cele mai populare la etapa actuală fiind POO. Conform Grady Booch, „tehnologia orientată pe obiecte a fost concepută în baza principiilor de abstractizare, încapsulare, modularitate, ierarhizare, tipizare, paralelilism şi persistență. Ceea ce este important constă în faptul că aceste principii se regăsesc în modelul orientat pe obiecte" [1]. Evoluția limbajelor de programare a cunoscut în timp apariția mai multor generații. S-a observat însă două tendințe:

1) Trecerea de la unități de programare mai mici la unele mai mari, care să ofere mai multe posibilități.
2) Dezvoltarea limbajelor de programare de nivel înalt.

Prima generație a limbajelor de programare o constituie perioada 1954-1958. Limbajele din această generație au fost concepute pentru a simplifica complexitatea formulelor matematice. Sunt caracteristice limbajele: FORTRAN I, ALGOL-58, Flowmatic şi IPL V.

A doua generație datează cu anii 1959-1961 şi se caracterizează prin concentrarea atenției asupra abstracțiunilor algoritmice. În acea perioadă calculatoarele au devenit mai puternice, ele fiind utilizate în soluționarea problemelor de ordin economic, de evidență a personalului etc. Sunt utilizate limbajele: FORTRAN II, ALGOL-60, COBOL, Lisp.

Odată cu apariția tranzistoarelor, costul unui calculator a scăzut considerabil; astfel, producerea PC a crescut exponențial. Creşterea PC a dus la apariția în perioada 1962-1970 a unei noi generații a limbajelor de programare. Au apărut limbaje precum: PL/I, ALGOL-68, Pascal, Simula. Aceste limbaje se caracterizează prin faptul că oferă un alt nivel de abstractizare a datelor, acordând programatorului posibilitatea de a crea noi tipuri de date.

Perioada 1970-1980 se caracterizează prin apariția mai multor limbaje de programare, care însă nu au făcut față cerințelor pieței. Problema privind abstractizarea obiectelor a fost caracteristică perioadei 1970-1980; astfel, a apărut limbajul Simula, iar în baza acestuia şi altele, precum Smalltalk, Object Pascal, C++, CLOS, Ada şi Eiffel. Aceste limbaje au fost numite orientate pe obiecte.

În momentul de față există foarte multe versiuni ale limbajelor de programare. Un rol aparte în evoluția limbajelor de programare se consideră a fi metodologia programării orientate pe obiecte. Această metodologie este bazată pe următoarele principii:

1. *Abstractizare* – proces care captează caracteristicile esențiale ale unui obiect, caracteristici ce diferențiază acest obiect de toate celelalte tipuri de obiecte. Prin abstractizare se rezolvă problemele cu un grad sporit de complexitate. Hoare susține: „Abstractizarea se manifestă prin determinarea similarității dintre anumite obiecte, situații sau procese din lumea reală şi în luarea deciziilor, ignorând pentru moment deosebirile dintre obiecte" [2].

După Seidewitz şi Stark, „există o serie de abstracțiuni de obiecte care pot să corespundă aproape exact realităților dintr-un anumit domeniu, iar pentru unele obiecte acestea nu au dreptul la existență" [3]. Aceste tipuri de abstracții includ:

- ✓ Abstractizarea entităților obiectului
- ✓ Abstractizarea comportamentului unui obiect
- ✓ Abstractizarea maşinii virtuale
- ✓ Abstractizarea arbitrară.

2. *Încapsulare* – proces mental executat pe o abstractizare în care elementele structurale ale abstractizării sunt izolate de cele ce constituie comportamentul său. Serveşte la separarea interfeței vizibile a unei abstractizări față de implementarea sa.

3. *Modularitate* – proces de divizare a unui sistem complex în părți (module) manevrabile. Apud Liskov, „modularitatea este nu altceva decât divizarea programului în mai multe unități, care, fiind compilate separat, în acelaşi timp comunică între ele" [4].

4. *Ierarhizare* – proces de clasificare pe niveluri de abstractizare. Un exemplu de ierarhizare este *moştenirea*, când corpul unei clase este transmis unei altei clase, fără copierea fizică a acestuia. Alt exemplu este *agregarea* sau obiecte care se includ în alte obiecte.

Acestea sunt principalele elemente care stau la baza modelului orientat pe obiecte. Dar, mai există şi:

5. *Tipizare* – reprezintă un mod de protejare a obiectelor. Prin intermediul tipizării se face distincție între obiectele diferitelor clase; astfel, obiectele nu pot fi schimbate sau, în unele cazuri, această schimbare este limitată.

6. *Paralelism* – este o proprietate, prin intermediul căreia se face distincție între obiectele active şi cele pasive.

7. *Persistență* – este proprietatea obiectelor care implică existența acestora şi după încetarea procesului care le-a creat. Starea obiectului şi codul corespunzător metodelor sunt memorate în baza de date. Tipurile obiectelor pot fi declarate persistente prin folosirea cuvântului-cheie persistent la momentul declarării, variabila fiind şi ea constrânsă la un tip persistent.

Un limbaj de programare se consideră a fi orientat pe obiecte dacă acesta suportă cel puțin primele patru principii. În caz contrar, dacă acesta susține cel puțin un principiu, el este considerat un limbaj bazat pe obiecte. Limbajele de programare orientate pe obiecte au evoluat considerabil. K.Schmucker prezintă evoluția acestora sub formă de arbore (Fig.1).

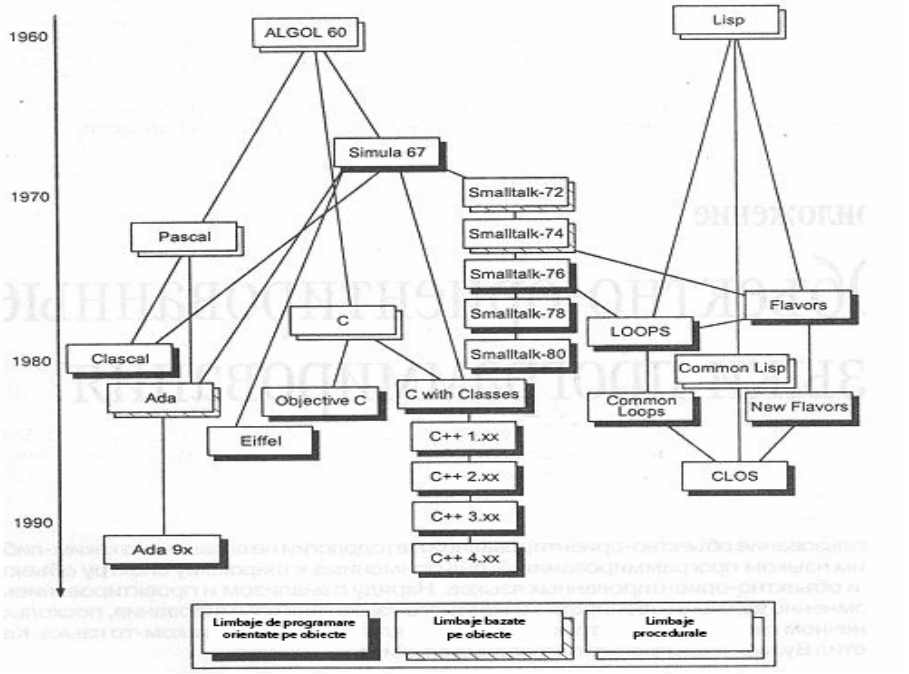

Fig.1. Evoluția limbajelor bazate pe obiecte şi orientate pe obiecte.

Conform planurilor de învățământ din cadrul universităților din Republica Moldova, cele mai populare limbaje de programare sunt limbajele Pascal şi C. Considerent din care, pentru predarea metodologiei orientate pe obiecte este mai convenabil a utiliza derivatele acestora: Object Pascal, elaborat de firma Apple Computer în 1986, de către un grup de cercetători în frunte cu Larry Tesler consultând şi fondatorul limbajului Pascal Niklaus Wirth; limbajul C++, care este conceput de către Bjarne Stroustrup. În 1979 apar diverse versiuni, cum ar fi „C with Classes", iar în 1983 apare limbajul C++. Din punctul de vedere al principiilor tehnologiei orientate pe obiecte, aceste limbaje includ în structura lor următoarele principii (Tab.1).

Tabelul 1

Principii	**Particularități**	**Object Pascal**	**C++**
Abstractizare	Instanțiere date	Da	Da
	Instanțiere metode	Da	Da
	Variabile de tip clasă	Nu	Da
	Metode de tip clasă	Nu	Da
Încapsulare	Date	public	public, protected, private
	Metode	public	public, protected, private
Modularitate	Tipuri de module	unit	fişiere
Ierarhizare	Moştenire	simplă	Simplă, multiplă
	Şabloane	Nu	Da
	Metaclase	Nu	Nu
Tipizare	Puternic tipizat	Da	Da
	Polimorfism	Da	Da
Paralelism	Multitasking	Nu	Indirect prin clase
Persistența	Obiecte persistente	Nu	Nu

Conform datelor din Tabelul 1, observăm că limbajul C++ include în structura sa mai multe principii; prin urmare, acesta ar fi mai indicat ca sursă în predarea POO.

Din 1990 în baza limbajelor POO s-au dezvoltat limbajele de programare vizuală. De o popularitate imensă se bucură limbajul Java. Asupra acestui limbaj au lucrat cercetătorii: James Gosling, Mike Sheridan şi Patrick Naughton în cadrul firmei Sun Microsystems, începând cu 1991 până la 23 mai 1995, când a fost lansată şi prima versiune a limbajului. Acest limbaj are foarte multe tangențe cu limbajul C++, dar, ca şi Object Pascal, nu suportă mecanismul moştenirii multiple.

Compania Borland lansează în 1995 prima versiune Borland Delphi 1.0. Această versiune a fost lansată pentru Windows 3.1 pe 16 biți. Borland Delphi are la bază limbajul Object Pascal. Aşa cum limbajul C++ se bucură de o popularitate imensă, compania Borland a lansat în 1997 platforma C++ Builder 1.0. Din 1997 şi până în prezent apar mai multe versiuni ale acestor platforme, care sunt asemănătoare după structura lor. La finele anului 2008 compania CodeGear lansează o nouă versiune RAD Studio, care include Delphi 2009 şi Builder 2009. În 2009, în cadrul pachetului RAD Studio, a fost lansată versiunea C++ Builder 2010.

Un alt limbaj de POO este şi limbajul C#. Acesta a fost dezvoltat de o echipă restrânsă de ingineri de la Microsoft, echipă din care s-a evidențiat Anders Hejlsberg (autorul limbajului Turbo Pascal şi membru al echipei care a proiectat Borland Delphi).

Tot de Limbaje de POO s-a preocupat şi compania Microsoft, care în 1997 a lansat Visual Studio, fiind prima platformă în care erau grupate mai multe instrumente de programare. În februarie 2002 Microsoft a lansat Visual Studio .NET. Programele dezvoltate în .NET nu sunt compilate în cod maşină (ca C++, de exemplu), ci într-un nou format, numit Microsoft Intermediate Language (MSIL). Visual Studio .NET 2002 a fost lansat în patru ediții: Academic, Professional, Enterprise Developer şi Enterprise Architect. Visual Studio 2008, care a fost lansat pe 19 noiembrie 2007, este concentrat pe dezvoltarea de aplicații Windows Vista şi Web. Conține un nou designer Windows Presentation Foundation şi un nou editor HTML. În schimb, J# nu este inclus. Visual Studio 2008 lucrează cu .NET Framework 3.5, dar poate susține şi aplicații create cu versiuni .NET Framework anterioare. În 2010 lansează Visual Studio 2010, NET Framework 4 şi Silverlight 4. .NET Framework este un model de programare puternic şi flexibil care permite dezvoltatorilor să conecteze desktop-uri multiple, dispozitive şi servere cu un epicentru securizat şi de încredere, care suportă multiple limbaje de programare, integrându-se cu instrumente şi tehnologii Microsoft şi non-Microsoft.

Silverlight 4 este elaborat pentru a ajuta companiile să creeze, să dezvolte şi să furnizeze aplicaţii interactive pentru Web, desktop şi dispozitive mobile.

Conform sistemului de învăţământ din Republica Moldova, de pregătirea profesorilor, în special a celor de la disciplinele exacte (precum Informatica) se ocupă instituţiile de învăţământ superior, principalele fiind: Universitatea de Stat, Universitatea Pedagogică „Ion Creangă", Universitatea de Stat din Tiraspol, Universitatea de Stat „Alecu Russo" din Bălţi.

Analizând planurile de învăţământ [5-7] ale universităţilor sus-menţionate, constatăm că la specialitatea „Informatica" sunt predate următoarele limbaje de programare (Tab.2).

Tabelul 2

Instituţia	Discipline conform planului de învăţământ	Numărul de credite
USM	POO în baza limbajului C++	4 credite
UST	Delphi	5 credite
	Programarea orientată pe obiecte	5 credite
UPS	Delphi	8 credite
	Java	3 credite
	Bazele de date (SQL, Delphi)	4 credite
USB	C++ Builder	4 credite
	Java	4 credite
	Visual Basic/Visual C	4 credite

Din punct de vedere profesional, al viitorului profesor de informatică, este foarte important ca acesta să cunoască cel puţin un limbaj de programare orientat pe obiecte. Predarea acestora trebuie efectuată din prisma tehnologiei orientate pe obiecte. Dacă el va cunoaşte elementele esenţiale care stau la baza acestei tehnologii, atunci va fi capabil să utilizeze diverse limbaje de programare orientată pe obiecte, nefiindu-i necesar să studieze diverse limbaje. După cum am observat, la moment acestea sunt foarte multe; respectiv, nu este posibil ca într-un curs sau două să fie studiate majoritatea limbajelor.

Concluzii

1. Actualmente există foarte multe limbaje de programare orientate pe obiecte. Deci, nu pot fi studiate toate limbajele. Având la bază tehnologia POO, este important a alege pentru instruire cel mai „apropiat" limbaj pentru studenţi.

2. Pentru studierea tehnologiei POO în instituţiile de învăţământ din Republica Moldova se recomandă a utiliza limbajele Object Pascal şi Borland Delphi pentru cei care cunosc limbajul Pascal şi, respectiv, C++ şi C++ Builder pentru cei care cunosc limbajul C. S-a observat că prin intermediul limbajelor companiei Borland studierea tehnologiei POO este asimilată mai uşor de către studenţi.

3. Pentru ciclul II de studii se recomandă a studia Platforma Visual Studio în baza limbajului C#, datorită posibilităţilor oferite de către această platformă.

Referinţe:

1. Grady Booch. Object-Oriented Design with Applications. Benjamin/Cummings, Redwood City, California, 2nd edition, 1994, p.25.
2. Dahl O., Dijkstra E. and Hoare C.A.R. Structured Programming. - London: Academic Press, 1972, p.83.
3. Seidewitz E. and Stark M. Towards a General Object-oriented Software Development Methodology. Proceedings of the First International Conference on Ada Programming Language Applications for the NASA Space Station. NASA Lyndon B.Johnson Space Center. TX: NASA, 1986, p.D.4.6.4.
4. Liskov B. A Design Methodology for Reliable Software Systems, in Tutorial on Software Design Techniques. Third Edition. - New York, NY: IEEE Computer Society, 1980, p.66.
5. http://www.upsc.md/_Plan_Studii/ITII/Informatica.html
6. http://www.usm.md/downloads/facultati/matematica/Plan-Informatica.pdf
7. http://bologna.ust.md/s_inf.htm

Prezentat la 21.11.2011