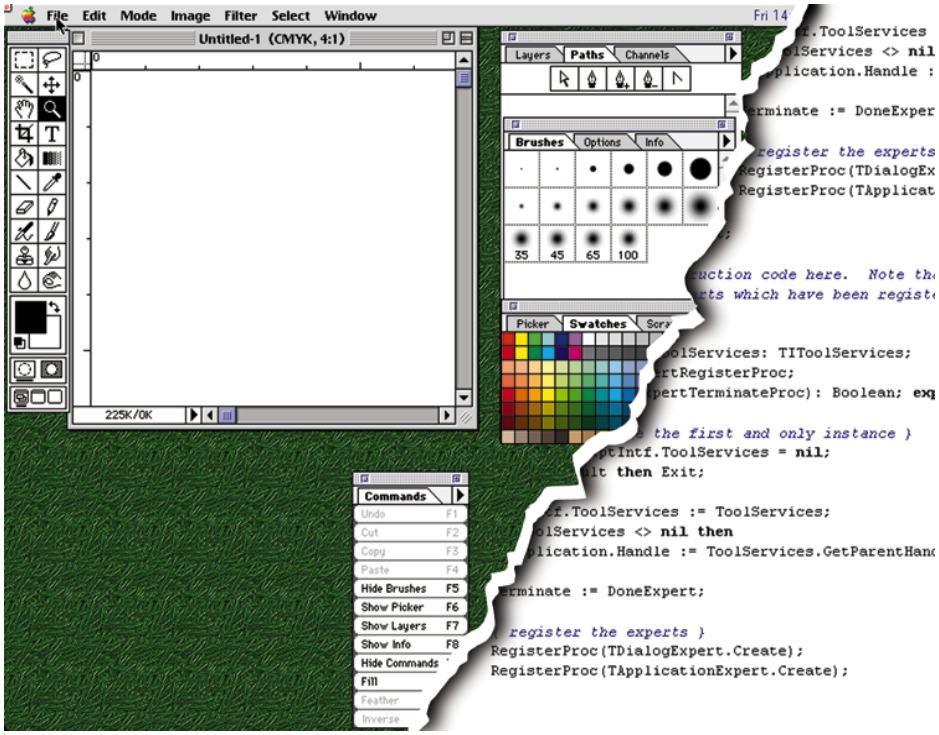


maciek glinka

Asystent, ekspert i kreator

Na początku były zera i jedynki, potem asembler, Fortran, Cobol i wiele innych języków programowania. Zdaniem niektórych, lata 50. i 60. w informatyce to już prehistoria. Nastała jednak nowa era – rozbudowanych „wizualnych" języków programowania, które w niczym nie przypominają swoich starszych braci.

Narzędzia pracy programistów w ciągu ostatnich lat przeszły wielką ewolucję jakościową. Pojawiły się nowe języki programowania, stare zostały znacznie zmienione lub powoli „wymarły". Dzisiejszy *Visual Basic* w niewielkim stopniu przypomina dawne dialekty Basica. Zasadniczo ulepszono Fortran, przez wielu już dawno skazany na wymarcie oraz język Cobol.

Rozwój języków programowania jest wymuszony przez rosnące wymagania rynku. Programować zaczynają nie tylko fachowcy; pakiety programistyczne coraz częściej trafiają „pod strzechy". Konsekwencją tego jest konieczność budowania oprogramowania znacznie prostszego w obsłudze. To z kolei komplikuje pracę ich twórców oraz wymaga wydajniejszych i bardziej niezawodnych narzędzi.

Od paru lat mamy do czynienia ze zintegrowanymi środowiskami programistycznymi (IDE – Integrated Development Environment). Integrują one kilka programów służących do budowania aplikacji. Należą do nich: kompilator, debugger, konsolidator (linker) i edytor.

Wczoraj...

Jednym z pionierów, wytyczających kierunki rozwoju współczesnych narzędzi programistycznych, był Visual Basic. Basic, przez wielu uważany za jeden z gorszych języków (jako wykorzystujący elementy programowania niestrukturalnego), we wrześniu 1991 roku ukazał się w nowej szacie, pozostawiono właściwie tylko starą nazwę. Visual Basic szybko zyskał zwolenników, przede wszystkim dlatego, że był narzędziem bardzo prostym. Umożliwiał, nawet mało wprawnemu użytkownikowi, stosunkowo łatwe tworzenie aplikacji dla Windows.

Alternatywnym rozwiązaniem stało się programowanie „okienkowych" aplikacji w języku C. Praca z ówczesnymi kompilatorami C była zajęciem żmudnym i stresującym. Wystarczy przypomnieć, że w pakietach C/C++ przeznaczonych dla Windows, debuggery pracowały w trybie znakowym. Korzystanie z nich (ciągłe przełączanie pomiędzy trybem graficznym i znakowym), nie należało, oględnie mowiąc, do najprzyjemniejszych.

Nowa koncepcja szybkiego programowania wizualnego (w którym trudny proces kodowania zastąpiony został przez proste ustawianie odpowiednich właściwości obiektów) szybko znalazła rzesze zwolenników. Microsoft podaje, że sprzedał 2 miliony kopii Visual Basica (od wersji 1.0), za pomocą których stworzono ok. 9–10 milionów aplikacji. Producenci innych języków programowania nie pozostali w tyle i szybko wypuścili na rynek „wizualne" wersje swoich kompilatorów. Wśród nich dominują kompilatory języka C++.

Na pierwszym planie: C++

Jest to najpopularniejszy język wykorzystywany przez producentów oprogramowania. Jego możliwości znakomicie predysponują go do tworzenia aplikacji dla środowisk „okienkowych".

Obecnie na rynku dominują cztery pakiety wykorzystujące C++. Są to: *Visual C++* (Microsoft), *Borland C++ Development Suite* (Borland), *Symantec C++* (Symantec) oraz *Watcom C++* (Sybase/Powersoft).

Ponieważ Visual C++ powstał w firmie, która stworzyła Windows, to wydaje się naturalne, że zyskał największe uznanie wśród twórców aplikacji dla tego systemu. Watcom C++ natomiast, to prawdziwy „kombajn"; za jego pomocą można budować aplikacje uruchamiane w różnych systemach operacyjnych. Ciekawostką jest, że kompilator ten potrafi generować moduły ładowalne (NLM) dla systemu Novell NetWare. Zdaniem wielu użytkowników, Watcom C++ generuje kod najbardziej zoptymalizowany, choć sam pakiet nie jest najprostszy w użyciu.

Bardzo ważna cecha Visual C++ to możliwość dodania do środowiska programistycznego (*Microsoft Developer Studio*) dodatkowych narzędzi, takich jak *Fortran PowerStation*, *Visual Test*, *Visual Source-Safe* oraz *Visual J++*. Pozwalają one na łączenie kodu napisanego w językach Fortran i C/C++ w tej samej aplikacji oraz wygodne wykorzystanie modułu

Visual Test do automatycznego testowania aplikacji. Visual SourceSafe ułatwia zaś kontrolę nad kolejnymi wersjami kodu, co jest szczególnie istotne w przypadku pracy grupowej.

Wszystkie pakiety oferują zbliżony zestaw narzędzi programistycznych i szereg mechanizmów automatyzujących pracę programisty (w tym automatyczne generowanie kodu obiektów). Pojawienie się jakiejś nowej funkcji w jednym z nich zwykle powoduje dołączenie bardzo podobnej opcji w najnowszej wersji pakietu konkurencji.

Wymienione programy pozwalają na automatyczne tworzenie szkieletu aplikacji, który stanowi deklarację indywidualnych wymagań użytkownika, np. obecność w aplikacji paska narzędzi i jego zawartość. Na podstawie tych informacji pakiet generuje bazowy kod aplikacji, który może być dowolnie modyfikowany i rozszerzany.

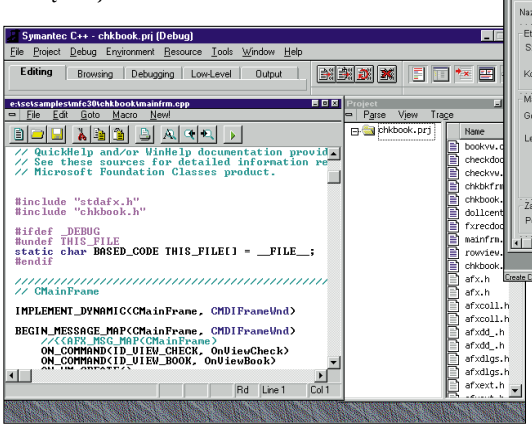
W tworzonym programie zwykle korzysta się z bibliotek funkcji i klas. Zdecydowanie dominuje tu biblioteka klas MFC (Microsoft Foundation Classes), której używa Visual C++ (co jest oczywiste), Symantec C++, Watcom C++ oraz – co dla wielu jest może zaskoczeniem – pakiet Borlanda. Ten ostatni przez wiele lat wraz ze swoim kompilatorem C++ dostarczał bibliotekę Object Windows Library (OWL), jednakże popularność MFC wśród programistów doprowadziła do tego, że ostatnio Borland doszedł z Microsoftem do porozumienia, w wyniku którego wraz z pakietem Borland Development Suite 5.0 dostarcza bibliotekę Microsoft Foundation Classes.

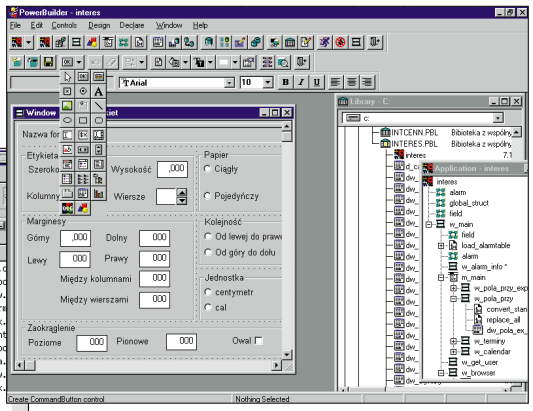
Korzystanie z takich bibliotek jak MFC czy OWL oraz koncepcji pracy oferowanych przez wymienione kompilatory C++ ma swoje wady i zalety. Programista nie może w dowolny sposób zmieniać wygenerowanego przez pakiet szkieletu aplikacji. Wszelkie zmiany muszą być zgodne z zasadami tworzenia aplikacji wymaganymi przez pakiet.

Kilka lat temu do pracy z kompilatorem C++ potrzebna była cała grupa dodatkowych narzędzi, poczynając od edytora, a kończąc na przyzwoitym debuggerze. Obecnie każdy pakiet IDE posiada wbudowany edytor. W produktach Borlanda jest to „okrojony" *Brief*; konkurencyjne środowiska zawierają podobne procesory tekstu. Pracując w środowiskach graficznych, wykorzystują wszystkie ich możliwości, (np. czcionki o różnych krojach, wielkościach i kolorach).

Dostarczane obecnie kompilatory mieszczą, poza dokumentacją, dziesiątki przykładowych aplikacji. Można je łatwo modyfikować i wykorzystywać do własnych celów. Ze względu na objętość dokumentacji, coraz częściej producenci narzędzi programistycznych dostarczają całe pakiety w postaci elektronicznej: z każdą wersją rośnie ona nie wolniej niż objętość samych pakietów. Na przykład w pakiecie Visual C++ 4.0 Standard Edition dokumentacja ma objętość ponad 3000 stron, zaś w wersji Professional jest znacznie obszerniejsza.

Do niedawna jedną z bardziej kłopotliwych czynności przy realizacji komercyjnych produktów było tworzenie dyskietek instalacyjnych. Obecnie zarówno Visual C++, jak i Borland C++ zawiera program *InstallShield Express*, który generuje programy instalacyjne i deinstalacyjne dla Windows 95 i Windows NT.





W pełni konfigurowalne paski narzędzi pozwalają programiście dopasować środowisko PowerBuildera stosownie do własnych potrzeb

Wyróżnianie różnych elementów kodu (np. słów kluczowych) innym kolorem stało się normą w wizualnych środowiskach programistycznych

Jako jedyny w omawianym towarzystwie, produkt Symanteca – C++ – nie dysponuje narzędziem do kontrolowania wersji kodu źródłowego. Możliwe jest natomiast dokupienie do niego (co dziwne tylko do kompilatora aplikacji 16-bitowych) zewnętrznego programu *PVCS Version Manager* opracowanego przez Intersolv, który, notabene, jest dostarczany wraz z pakietem Borland C++.

Ciekawym rozwiązaniem, zastosowanym przez firmę Borland, służącym do modyfikowania środowiska pracy, jest język Script. Za jego pomocą użytkownik może zmieniać dowolne elementy środowiska IDE lub dodawać nowe. Symantec C++ 7.2, podobnie jak Visual C++, pozwala natomiast tworzyć elementy kontrolne zgodne ze standardem ActiveX.

Symantec zastosował w swoim pakiecie kilka ciekawych rozwiązań. Jednym z nich jest program *Graphics Data* znany z debuggera *Multiscope* (pierwszego „odpluskwiacza" pracującego w trybie graficznym). Aplikacja ta pozwala na graficzną prezentację złożonych struktur danych. Natomiast w Borlandzie C++, poza debuggerem, bardzo przydatny jest również *CodeGuard 32/16* – program przechwytujący komunikaty generowane przez system po wykryciu próby dostępu do niedozwolonego obszaru pamięci. Aplikacja ta rejestruje wykryte błędy i wskazuje fragmenty kodu odpowiedzialne za ich wystąpienie.

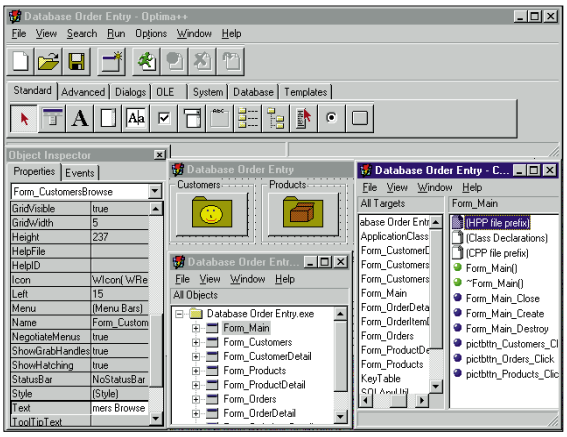
Słowniczek

debugger – program, pozwalający na krokowe uruchamianie i testowanie aplikacji
enkapsulacja – możliwość zawarcia w jednej strukturze danych pól oraz funkcji i procedur (tzw. metod), wykorzystywanych do operacji na polach klasy
klasa – struktura danych, zawierająca zarówno pola, jak i metody
kompilator – narzędzie tłumaczące kod źródłowy napisany w dowolnym języku programowania (np. Pascal, C++, asembler) na kod maszynowy
konsolidator (linker) – program łączący kod wynikowy, otrzymany w wyniku procesu kompilacji, z bibliotekami funkcji; wynikiem jego działania jest wygenerowanie kodu wykonywalnego (EXE)
obiekt – pojedynczy egzemplarz klasy
polimorfizm – dokonywanie wyboru metody, która ma zostać wywołana w danym punkcie programu, nie na etapie kompilacji tej aplikacji, lecz dopiero w trakcie jej wykonywania

Jeśli nie C, to co?

Jeżeli ktoś nie jest miłośnikiem języka C++, może korzystać z takich programów jak *Visual Basic*, *PowerBuilder* czy *Delphi*. Ostatni z wymienionych pakietów wdarł się przebojem na rynek wiosną ubiegłego roku i od razu zdobył uznanie wśród programistów. Wszystkie wymienione produkty są przeznaczone do szybkiego tworzenia aplikacji klient-serwer. Trudno jednak stwierdzić, który z nich jest najbardziej popularny. W aplikacjach bazodanowych liderem jest *PowerBuilder*, ze względu na fakt, że jego producent – firma PowerSoft – jest twórcą bardzo dobrego serwera baz danych, znanego pod nazwą *Sybase SQL Server*.

Najnowsza wersja *PowerBuilder Enterprise 5.0* zawiera wiele nowych funkcji i udoskonaleń. Przede wszystkim tworzy 32-bitową aplikację skompilowaną do kodu maszynowego (dla Windows 95 i Windows NT), dzięki czemu tworzone za jego pomocą programy są szybsze. W pakiet wbudowany jest kompilator oparty na Watcom C++. Zmodyfikowano też lokalną bazę danych *Watcom SQL*, która obecnie nosi nazwę *Sybase SQL Anywhere*.



Środowisko Optima++ przypomina pakiety Delphi oraz Visual Basic

PowerBuilder zawiera narzędzia wspomagające pracę zespołową. Służy do tego moduł *ObjectCycle*. Za jego pomocą programista wybiera obiekty z bazy, kontroluje ich wersje i tworzy raporty o stanie ich wykorzystania. Bardzo interesującym elementem pakietu PowerBuilder Enterprise jest program *InfoMaker*, który pozwala na tworzenie lub modyfikowanie raportów, analiz i zestawień z baz danych. Jest to program bardzo łatwy w użyciu. Do realizacji raportów używa język SQL, jednakże jego znajomość nie jest wcale konieczna do pełnego wykorzystania programu. Procedury SQL powstają automatycznie (niejako w tle) w trakcie graficznego konstruowania zapytań do bazy danych.

Wykorzystywany w PowerBuilderze język skryptowy nie budzi zachwytu. Wielu profesjonalnych programistów preferuje „przyzwoity" język obiektowy, np. C++. Kiedy na rynku pojawiło się Delphi ze swoim w pełni obiektowym Pascalem, PowerSoft zdecydował się na stworzenie konkurenta dla Delphi. I takim konkurentem ma być program *Optima++*, oferujący programiście większość zalet PowerBuildera z wygodnym środowiskiem pracy, a z drugiej strony opierający się na nieformalnym standardzie C++.

Optima++ zasadniczo przeznaczony jest do tworzenia aplikacji klient-serwer. Jako efekt swej pracy programista otrzymuje w pełni skompilowany program (EXE) działający w środowisku Windows 95, Windows NT i Windows 3.11 (z wykorzystaniem Win32). Jak podaje producent, Optima++ to pakiet dla zaawansowanych programistów wykorzystujących języki: Pascal, Basic, C i C++ i chcących zwiększyć efektywność swojej pracy. Ważne jest, że w pracy z Optimą++ można używać istniejącego kodu w C++. Oczywiście nie zawsze ma to sens. W pakiecie *Optima++ Developer* poza obfitą dokumentacją (ponad 900 stron) użytkownik ma do dyspozycji kilkadziesiąt przykładów, które może bez problemów włączać do swoich aplikacji.

Wizualna podstawa

Microsoft szczególną wagę przykłada do rozwoju języka Visual Basic. Aktualna wersja 4.0 jest znacznie poprawiona, jeżeli chodzi o tworzenie aplikacji klient-serwer, przetwarzanie rozproszone i pracę grupową. Co prawda pakiet dalej nie posiada „prawdziwego" kompilatora (program kompilowany jest do kodu pośredniego) i malkontenci zżymają się, że aplikacje stworzone za jego pomocą działają za wolno. Dostęp do baz danych odbywa się za pomocą 32-bitowego mechanizmu Jet 3.0, wyraźnie udoskonalonego w porównaniu z poprzednimi wersjami. Warto zwrócić uwagę, że to samo jądro (ang. engine) obsługi baz danych znajdujemy w programie *Access 95*.

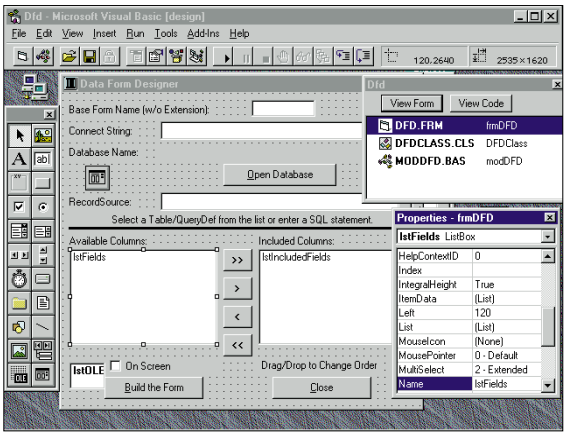
Największym przełomem w wersji VB 4.0 jest automatyzacja OLE (tzw. OLE automation objects). Niestety, Visual Basic wciąż nie jest systemem obiektowo zorientowanym. Nie można tworzyć nowych klas obiektów na podstawie już istniejących, nie można też bazować na takich obiektach, jak formularze. Pakiety konkurencyjne natomiast, np. Delphi, robią to od dawna. Jednego można być pewnym – Visual Basic na pewno będzie się rozwijał i warto znać jego możliwości.

Visual Basic czy Delphi?

Tymczasem Delphi firmy Borland ma się całkiem dobrze. Program ten miał wyprzeć z rynku Visual Basic. Czy mu się to udało? Trudno jest to jednoznacznie stwierdzić; na pewno przejął część jego

użytkowników. Wersja 1.0 została życzliwie przyjęta przez programistów. Połączenie łatwości i szybkości tworzenia aplikacji z w pełni obiektowym językiem programowania dało bardzo dobre efekty. Wersja 2.0 nie była już takim hitem rynkowym, a wynikło to najprawdopodobniej z błędów marketingowych Borlanda (Delphi 2.0 nie potrafi tworzyć aplikacji 16-bitowych).

Zaletą Delphi jest bardzo duża szybkość kompilacji. W skład pakietu *Delphi Client/Server Suite 2.0* wchodzi lokalny serwer baz danych SQL o nazwie *InterBase*. Oprócz niego programista otrzymuje szereg narzędzi pozwalających konstruować bazy danych i przetwarzać zawarte w nich dane. SQL Explorer umożliwia przeglądanie baz danych i towarzyszących im definicji. Pakiet wyposażono w sterowniki, za pomocą których użytkownik bez problemu może przetwarzać bazy danych zapisane w formacie: InterBase'a, Oracle'a, Informixa, Microsoft SQL oraz Sybase'a.



Wygląd środowiska Visual Basic 4.0 w stosunku do poprzednich wersji pozostał w zasadzie nie zmieniony

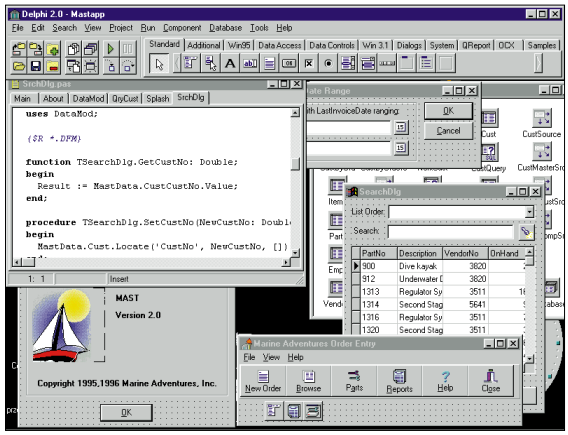
Delphi zawiera debugger kodu źródłowego, dzięki któremu uruchamianie programu staje się łatwiejsze. Jest on bardzo podobny do „odpluskwiaczy" kompilatorów C++. Przy tworzeniu rozbudowanych aplikacji klient-serwer bardzo przydatnym narzędziem jest *SQL Monitor*. Program ten zarządza komunikacją pomiędzy bazą danych a Delphi i ułatwia optymalizowanie efektywności pracy bazy danych. Podobnie jak Borland C++, Delphi zawiera moduł *PVCS Version Control Manager* (program do kontroli wersji programów źródłowych) oraz *InstallShield Express* – program do tworzenia dyskietek instalacyjnych dla Windows 95 i Windows NT.

Aby ułatwić programistom tworzenie aplikacji bazodanowych, generator raportów z baz danych Delphi wyposażono w program o nazwie *Report Smith*. Jest to aplikacja, za pomocą której można budować różnego rodzaju raporty składające się np. z wykresów. Program ten może być uruchomiony z „wnętrza" stworzonej za pomocą Delphi aplikacji albo jako samodzielny moduł.

Przebój sezonu – Java

Java to temat wielu rozmów informatyków. Ten najnowszy język programowania stworzony w firmie Sun jest ściśle związany z Internetem, jedną z najdynamiczniej rozwijających się gałęzi rynku komputerowego. Do niedawna jedynym źródłem informacji o Javie i narzędziach do programowania w tym języku była firma Sun. To właśnie stąd (*www.javasoft.com*) miłośnicy Javy ściągali pakiet *Java Development Kit* zawierający podstawowe narzędzia i masę przykładowych programów.

Wielkie firmy software'owe nie dały długo na siebie czekać. Pierwszy był Symantec ze swoim pakietem *Café*. Osobom korzystającym z Symantec C++ wyda się on znajomy, gdyż jest zbudowany na podstawie firmowego środowiska IDE, w pełni zgodnego z Java Development Kit i zawiera nawet oryginalne przykłady z JDK.



Tworzenie aplikacji za pomocą Delphi, ze względu na dużą funkcjonalność i niezawodność pakietu, jest zajęciem prostym i przyjemnym

Środowisko programistyczne Symantec Café 1.0 jest bardzo podobne do oferowanego przez Symantec C++. Edytor tekstu umożliwia wyróżnianie kolorami różnych elementów kodu źródłowego, np. słów kluczowych; dostępny jest także graficzny debugger pracujący na poziomie kodu źródłowego. Do szybkiego generowania projektów służą moduły *Project Express* i *AppExpress*. Dużą zaletą pakietu jest szybkość kompilacji, która jest kilka razy większa od szybkości oryginalnego kompilatora z JDK.

Microsoft i Borland nie pozostają w tyle. Ten pierwszy właśnie prowadzi intensywne prace nad programem *J++*, który jest oczywiście kompilatorem Javy. W chwili obecnej produkt przechodzi testy beta. Pakiet jest zintegrowany z programem Developer Studio, dostarczanym przez Microsoft. Borland natomiast kończy konkurencyjny produkt o nazwie *Latte*. W pakiet wbudowana będzie obsługa zapowiadanej przez SunSoft specyfikacji Java Beans, odpowiednika ActiveX.

Quo vadis?

Jaka będzie przyszłość narzędzi programistycznych? Gdyby ekstrapolować niektóre ich dzisiejsze parametry, to przyszłość może wyglądać czarno. Pakiety osiągają monstrualne rozmiary, z ogromną dokumentacją, której większość użytkowników nie będzie w stanie przeczytać. Około 10 lat temu kompilator *MSC 3.0* wraz z linkerem oraz niezbędnymi bibliotekami zajmował niecałe dwie dyskietki o pojemności 360 KB. Na dyskietkach mieścił się jeszcze edytor oraz prosta aplikacja. Obecnie Visual C++ 4.2 dostarczany jest na trzech CD-ROM-ach.

Niestety, programy tworzone za pomocą współczesnych narzędzi wcale nie są proporcjonalnie lepsze od tych starych. Nowe systemy operacyjne posiadają coraz bogatsze możliwości, co stawia większe wymagania przed programami narzędziowymi.

Zapewne pojawią się nowe narzędzia programistyczne, które jeszcze bardziej uproszczą tworzenie aplikacji. Niektóre stare języki programowania przejdą, wbrew przewidywaniom niektórych, kolejną metamorfozę. Najlepszym przykładem może być język Cobol. Wg danych Datapro Information Services Group, na całym świecie funkcjonuje około 150 miliardów linii kodu źródłowego w Cobolu (głównie w aplikacjach o kluczowym znaczeniu dla świata biznesu). Wartość ta jest zwiększana każdego roku o kolejnych 5 miliardów. Zaproponowano już standard COBOL '97, który obejmuje obiektowe elementy języka takie, jak definicje klas, tworzenie podklas, enkapsulację danych i polimorfizm.

Janusz Żmudziński