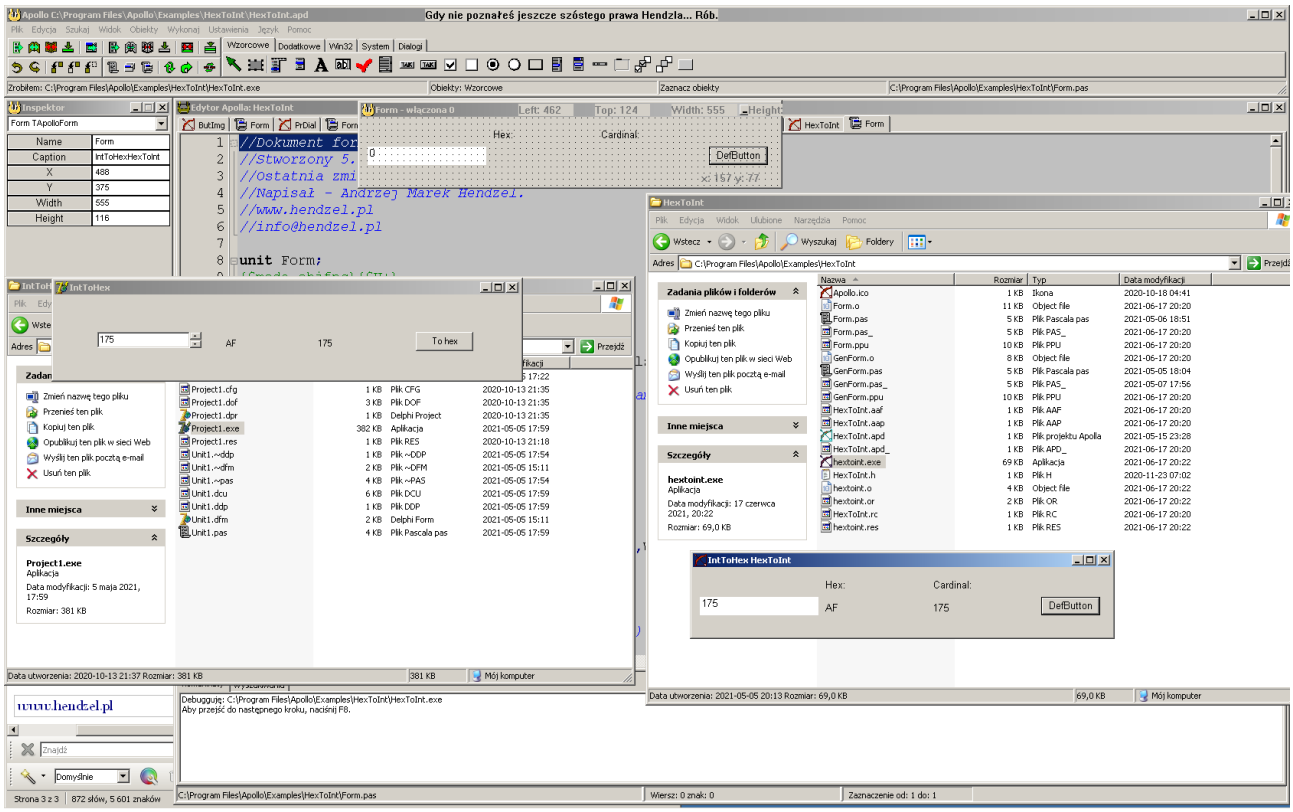
Apollo powstał w Apollu

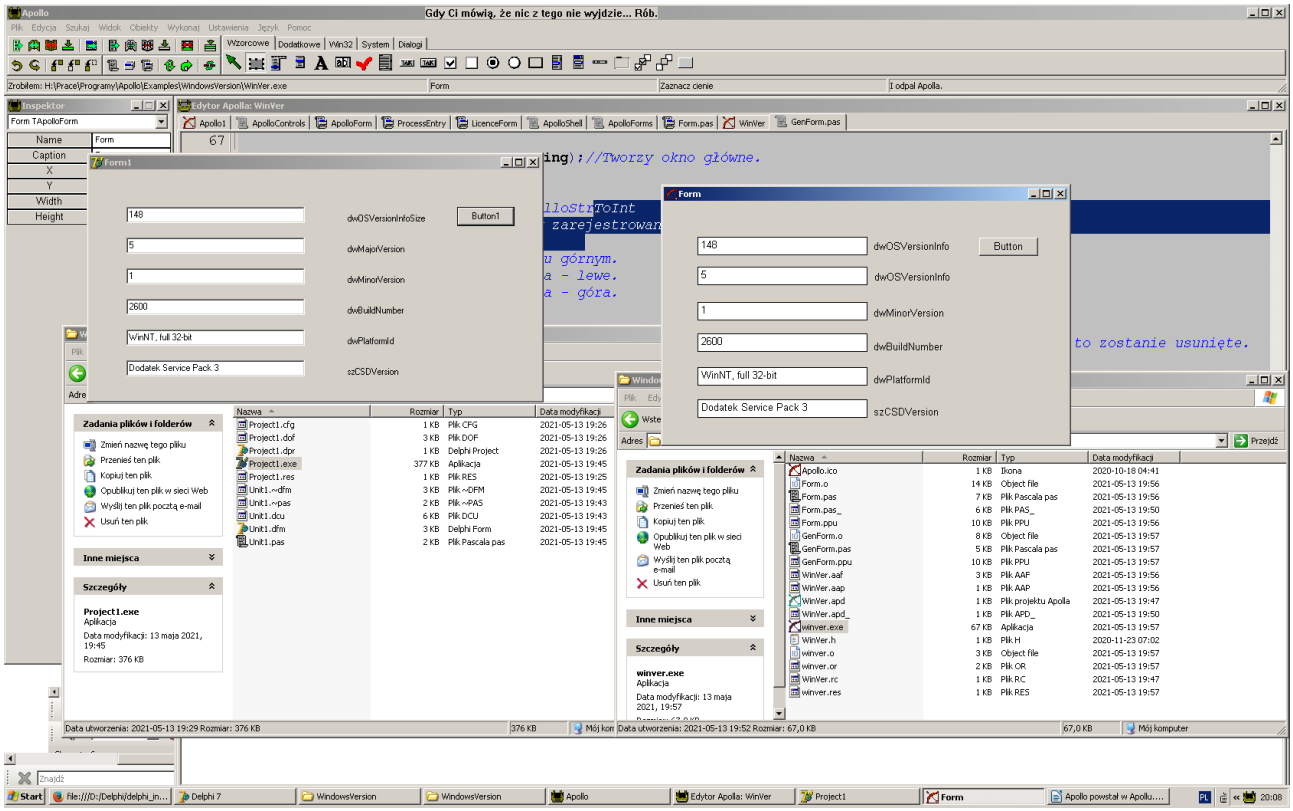
Tak, to możliwe. Apollo był najpierw zwykłym edytorem tekstu zaczerpniętym z programu Dev-Pas. Kompilował się z pliku BAT (*.bat). Po dodaniu modułu uruchomieniowego i prostego przycisku w menu Kompiluj, zaczął się sam kompilować. I tak powstał zalążek, z którego krok po kroku, po każdej zmianie źródła otworzonego i wyedytowanego w Apollu, powstawał stale nowy Apollo. Można powiedzieć, że rósł jak kula śniegowa, albo lepiej jak żywa istota, rozbudowywany i stale ulepszany.

Są ludzie wiecznie niezadowoleni a szczególnie z osiągnięć innych. A gdyby tak, zamiast szukać dziury w całym i krytykować to, że czegoś nie ma, zauważyliby to, co jest. Tacy ludzie są jak wsiadający do samolotu, który narzeka, że mu tu okna otworzyć nie można, a w samochodzie może. Może i w karecie. A furą końską jak się jeździ z rozwianymi włosami. Jednakże chyba nie z takimi prędkościami i nie na takich wysokościach, jak zwykłe dziś samoloty pasażerskie, gdzie udusić się można bez powietrza, gdyby nie hermetyzowana kabina.



Ale my zauważmy, że Apollo wyrzuca z siebie aplikacje o bardzo małych rozmiarach. Są to w istocie aplikacyjki. Na rysunku widać takie maleństwo z Apolla do odwracania zmiennych całkowitych w szesnastkowe i na odwrót oraz prawie identyczny programik z Delphi. **Źródło procedur programu zostało zaimportowane z Apolla do Delphi.** Program z Apolla ma 69 KB a delficki (czy delphicki) ma 382 KB. Nie powiemy, jaki jest plik lazarusowy odpowiedniej aplikacji - co niech sobie Czytelnik sam sprawdzi. A przecież Apollo używa (jako silnika) tego samego (a czasem ciut nowszego) kompilatora FPC co Lazarus. Nasze maleństwo z Apolla ma nawet o jeden komponent wizualny więcej niż jego delficki odpowiednik a mimo to rozmiary na dysku są prawie sześciokrotnie mniejsze. Taka aplikacja szybciej się kompiluje, startuje, bo i nośnik (twardy dysk, pendrive... czy pendrajw) załaduje ją szybciej i skakania po RAM-ie jest mniej, czyli i chodzi szybciej. Czy to mało?

Na następnym rysunku są dwa identyczne funkcjonalnie programy Apolla (67 KB) i Delphi (377 KB). Oba podają wersję systemu Windows.



To, że Apollo nawyrzuca więcej plików dodatkowych, nic nie zmienia, gdyż praktycznie wszystkie te pliki (oprócz *.rc, *.h i *.ico) można usunąć z katalogu, po skończeniu pracy nad projektem, gdyż Apollo je odbuduje. Ma tylko jeden plik źródła więcej, zazwyczaj nazywany `GenForm`, który zawiera klasę `TApollo` i funkcje eksportowe generujące pętlę karuzeli Windows `WindowProc()` i rejestrację klasy aplikacji `WinRegister()`.

Dzięki temu plikowi programista wie, co aplikacja naprawdę robi. Czy to ciągle mało?

Są tacy, dla których mało to za mało? Zatem, dołóżmy im więcej. W Apollu jest na przykład funkcja `HexToInt()`. Ma nieco mylącą nazwę, bo w istocie zamienia wartości szesnastkowe na zmienne naturalne a nie ogółem na całkowite, ale robi to sprawnie. Dodatkowo wartości wpisane w okienko do zamiany najpierw na zmienną szesnastkową, mogą być z wtrąceniami spoza cyfr a mimo to Apollo przekształci je prawidłowo w liczbę i zapisze w hex-ie (czy heksie). Jak kto woli. I to samo zrobi funkcja `HexToInt()`. Na przykład z formatu $00AF, czy #00AF, czy 00AF czy ostatecznie AF zamieni na prawidłową wartość liczby naturalnej 175. To wyjątkowo plastyczna metoda. Nie zwraca błędów z byle powodu jak delfickie czy lazarusowe funkcje.

Apollo ma sporo takich funkcji. Nie wszystkie są w pełni wypróbowane, mogą zwracać jakieś błędy. Jednakże obecnie Apollo to program w wersji Hiper Beta... o czym jest na stronie www.hendzel.pl i w samym Apollu. A sama aplikacyjka hexowa leży w katalogu (choć większość niezadowolonych woli nazwę folder) Examples/HexToInt. Co widać na obrazku. Czyli dostarczana jest z Apollem jako przykład w postaci kodu źródłowego projektu wraz z wieloma innymi przykładami.

Apollo nie zapisuje domyślnie projektu ani żadnego pliku wraz z kompilacją – czyli przed nią. Kompilator pobiera wartości do kompilacji zawsze z pliku źródła, zatem łatwo się domyślić, że musi być zapisane źródło przed kompilacją, by uwidocznić wynik zmiany źródła. Czemu tego nie robi? Aby programiście weszło w krew, że swoją pracę trzeba zapisywać w trakcie pracy, by jej nie utracić. Automat rozleniwia. I czasami pracując klepie się kod, licząc na to, że guzik kompilacji załatwi wszystko i zapisze plik źródła. A tu bęc – i wyłączyli prąd, czy poszedł akumulator w UPS-ie czy laptopie. Ktoś, kto nigdy nie stracił wyników swojej pracy, nie wie, co to jest. Dlatego lepiej mieć odruch – jest coś ważnego wklepane, zapisuję plik. To jak używanie kierunkowskazu. „E, po co mam się wysilać, kiedy pusto.” - i tak stale. Aż tu nagle pakuje się taki w duży ruch uliczny i znowu (tylko może trochę większe) bęc. Bo się przyzwyczaił, że mu środowisko samo zapisze plik źródła. Czy ktoś myśli, że to dużo pracy, dokleić zapis przed kompilacją? To dwadzieścia sekund pracy. Ale zły nawyk, to zły nawyk, a my chcemy mieć dobre nawyki.

Jest wiele takich miejsc w Apollu. I dzięki temu środowisko jest użyteczne i szybkie. Ma małą paletę komponentów, ale tu trzeba poczekać. Nie pali się. Najważniejsze są mechanizmy pracy, analizy kodu i jakości kompilacji. I są pilniejsze potrzeby niż budowanie palet. Na paletach jedzie obecnie pół Polski, bo wygląda to tak, że nic innego nie umiemy, tylko zbijać palety i potem na nich wozić cudze towary. I to u obcych na placu. A to nie tylko nieamerykańskie, ale przede wszystkim niepolskie.

Z Bogiem.

Andrzej Marek Hendzel