2. Przydatne narzędzia i początki programowania.

W kolejnych częściach artykułu będę starał się przedstawiać proste programy i narzędzia, które są pomocne przy tworzeniu plikowych programów, zgodnych z DOSem. Oznacza to, że programy tu przedstawione będą poddawane dalszej obróbce, takiej jak kompilowanie, linkowanie, łączenie z danymi, itp.

Zacznijmy od przygotowania sobie środowiska pracy. Potrzebne nam będą: edytor (czyli właściwy Turbo Basic XL), kompilator do niego (Turbo Basic Compiler), linker i runtime (czyli biblioteka do uruchomienia skompilowanych programów). Z dodatkowych rzeczy osobiście używam Super Packera – którym nadaje nagłówki binarne danym – oraz programu Mossad do linkowania całości (często moje programy nie wchodzą do standardowej pamięci Super Packera). Do tego przydatne bywają dowolne programy graficzne, edytory fontów, muzyki (koniecznie z playerami), ale o tym później.

Aby zapoznać się z tworzeniem programu proponuję zacząć od standardowego w tym przypadku programu typu „hello World!":

```
10 ? CHR$(125)
20 ? "PROGRAM TESTOWY"
```

Zapisujemy go na przykład pod nazwą TEST.TB (SAVE"D:TEST.TB"). Oczywiście, ten programik nie korzysta z dobrodziejstw TB XL, ale pozwoli nam zapoznać się z kolejnymi etapami przygotowania programu plikowego uruchamianego pod DOSem.

Teraz przechodzimy do DOS-u i uruchamiamy kompilator (L => TBC.COM). Jego obsługa jest banalna: numer napedu wybieramy poprzez wciśnięcie cyfry odpowiadającej numerowi napędu, w jakim mamy nasz plik źródłowy (w przykładach zawsze będzie to napęd numer 1), a następnie wybieramy za pomocą strzałek nasz plik źródłowy. Jeśli plik nie jest prawidłowym zbiorem TB XL – zostaniemy o tym powiadomieni stosownym komunikatem, zaś jeśli wszystko jest o.k. - automatycznie zacznie się kompilacja programu.

Jeśli wszystko przebiegnie poprawnie – zostaniemy poproszeni o wpisanie nazwy skompilowanego programu, przy czym automatycznie zostanie dodana końcówka pliku .CTB, mówiąca o tym, że jest to skompilowany plik TB XL. Potem zostaniemy spytani, czy chcemy wykonać kolejną kopię skompilowanego programu – dla nas jest ona zbędna.

Jeśli w programie wystąpi jakikolwiek błąd – proces kompilacji zostanie przerwany, a na ekranie pojawi się stosowny komunikat wraz z numerem li-

nii w której wystąpił oraz numerem błędu. Warto go sprawdzić, bo często drobna sprawa niweczy godziny pracy. Zdarza się także, że program jest za długi i nie mieści się w buforze kompilatora.

Zakładając, że wszystko poszło dobrze – wracamy do naszego DOS-u (control-D) lub powtarzamy kompilację dla kolejnych programów.

Przyszła pora na zlinkowanie naszego dzieła z runtime-m (używamy poprawionego pliku RUNTIME2.EXE. Możemy tego dokonać funkcją append DOSu, skorzystać z dowolnego programu linkującego – lub najlepiej – skorzystać z dedykowanego linkera, co zazwyczaj czynię. Tu mała dygresja: linker może się znajdować w dowolnym napędzie, jednak plik RUNTIME2.EXE musi się znajdować w napędzie D1:. Po poprawnym wczytaniu linkera zostaniemy poproszeni o podanie pliku źródłowego – podajemy pełną ścieżkę (na przykład: D:TEST.CTB) oraz docelowy (D:TEST.XEX). Po poprawnym zlinkowaniu koniecznie należy wcisnąć klawisz ESC – błąd w kodzie linkera powoduje, że nie zamyka on w innym przypadku poprawnie plików! Po wciśnięciu ESC pojawia się standardowe wyjście z skompilowanego programu TB XL – wybieramy D (wyjście do DOS-a).

Możemy teraz sprawdzić, czy nasz programik działa pod DOS-em: wybieramy L i ładujemy plik TEST.XEX. Jak widać można (tylko to paskudne zakończenie programu... Jak to można obejść – można będzie przeczytać w dziale TB XL lub Kruczki i Sztuczki już niedługo.

Przed kompilacją należy pamiętać, że niektóre dane programu są umieszczone w innych miejscach niż w programie źródłowym. Rodzi to czasem pewne problemy techniczne, gdy okazuje się, że programy źródłowe działają w zgoła inny sposób od skompilowanych (rzadki przypadek, ale się zdarza). Zazwyczaj pomaga poprzesuwanie danych binarnych używanych w naszym programie głównym (na przykład grafika, muzyka, itp.), choć trzeba się liczyć z tym, że wtedy nasz program może przestać działać należycie w wersji źródłowej. Jednak zdarza się to stosunkowo rzadko, a pewne nawyki i upodobania można nabyć wraz z nauką programowania, o czym będzie w dalszej części programu.

Należy się także wystrzegać dynamicznie wyliczanych linii w naszych dziełach – co jest jedną z przyczyn niemożności skompilowania niektórych programów (na przykład oryginalne KOLONY).

Z uwagi na Runtime należy też pamiętać o tym, aby zwrócić uwagę na obsługę niektórych wrażliwych miejsc pamięci – o czym będzie w dalszych częściach artykułu.