Wstęp

Książka obejmuje ćwiczenia z programu Borland Turbo Pascal 7.0 i przeznaczona jest dla czytelników, którzy dopiero zaczynają zgłębiać tajniki programowania.

W pierwszym rozdziale omówione są podstawowe zagadnienia obsługi systemu DOS i Windows - związane z uruchomieniem języka Turbo Pascal.
UWAGA: Programy napisane dla systemu DOS (np. Borland Turbo Pascal 7.0) można również uruchamiać z poziomu systemu Windows.

W drugim rozdziale omówione jest środowisko programistyczne.
Następny rozdział opisuje podstawowe konstrukcje tego języka.
W czwartym rozdziale przedstawione są programy na średnim poziomie trudności.
Piąty rozdział prezentuje pisanie programów bardziej złożonych w stosunku do programów przedstawionych w poprzednich rozdziałach.
W ostatnim rozdziale umieściłem ćwiczenia do samodzielnego wykonania.

UWAGA: Ze względu na systemowe kłopoty z polskimi znakami diakrytycznymi, nie są one używane przy pisaniu kodów poszczególnych programów.

Po zakończeniu ćwiczeń przedstawionych w tej książce, namawiam Cię czytelniku do kontynuowania nauki w języku Borland DELPHI. Kontynuując naukę programowania możesz wykorzystać moje książki pt. Ćwiczenia z... „Podstawowych komponentów Delphi”, „Delphi 7”, „Tworzenie prostych programów użytkowych w Delphi” oraz „Pisanie programów generujących strony WWW w Delphi”, wydanymi w Wydawnictwie MIKOM. Język DELPHI umożliwia wizualne projektowanie aplikacji, dzięki czemu zaliczany jest do narzędzi typu RAD (Rapid Application Development) – co oznacza szybkie tworzenie aplikacji. W tym języku tworzy się bardzo duże aplikacje dla systemu Windows. Język ten jest również dostępny dla systemu Linux.

Wszystkim czytelnikom życzę owocnej pracy.

Autor

1. Uruchomienie języka Turbo Pascal

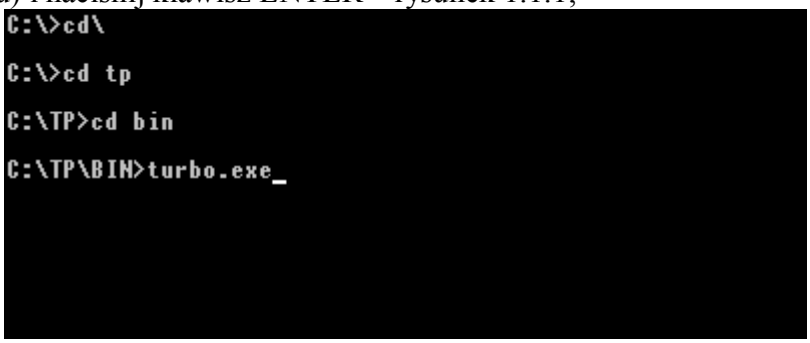
W rozdziale tym przedstawione są ćwiczenia, mające na celu przypomnienie sposobów uruchamiania programu Borland Turbo Pascal.

Ćwiczenie 1.1. Uruchomienie programu Turbo Pascal z linii poleceń

Uruchom program Turbo Pascal z linii poleceń.

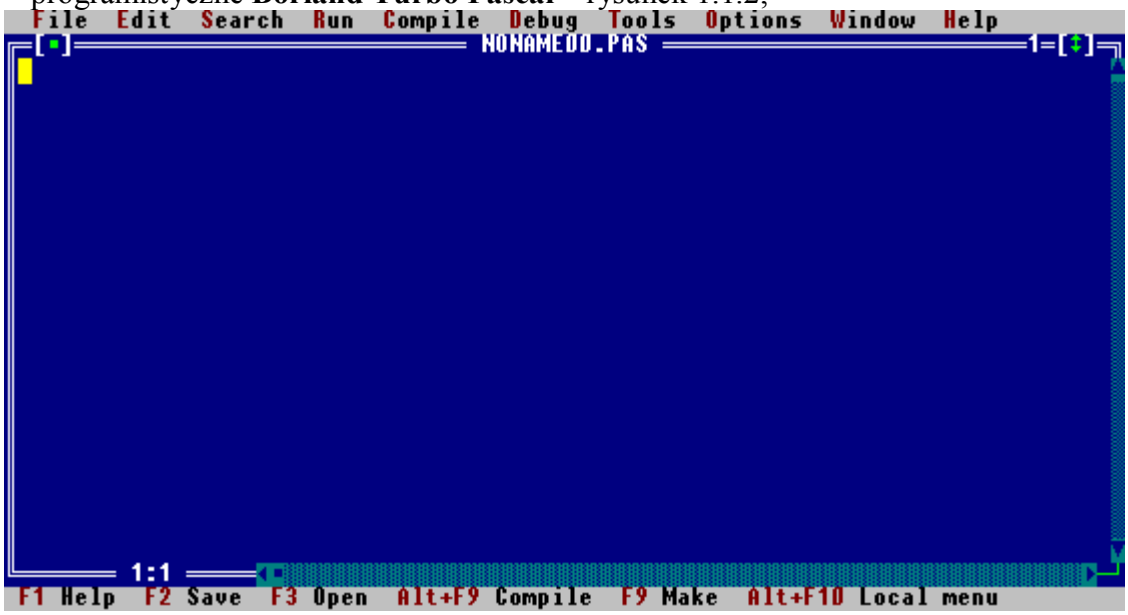
Sposób wykonania

- W linii poleceń wpisz polecenie **CD** (polecenie to umożliwia przejście do katalogu głównego, jeżeli aktualnym katalogiem jest inny katalog np. C:\NC\DOC) i naciśnij klawisz ENTER;
- Wpisz polecenie **CD TP** (przechodzi do katalogu TP) i naciśnij klawisz ENTER;
- Wpisz polecenie **CD BIN** (przechodzi do katalogu BIN) i naciśnij klawisz ENTER;
- Wpisz polecenie **TURBO.EXE** lub **TPX.EXE** (w zależności jaki plik masz w posiadaniu) i naciśnij klawisz ENTER – rysunek 1.1.1;



Rysunek 1.1.1. Widok kolejno wpisanych rozkazów

- Po wykonaniu wyżej wymienionych czynności zobaczysz zintegrowane środowisko programistyczne **Borland Turbo Pascal** – rysunek 1.1.2;



Rysunek 1.1.2. Widok zintegrowanego środowiska Borland Turbo Pascal

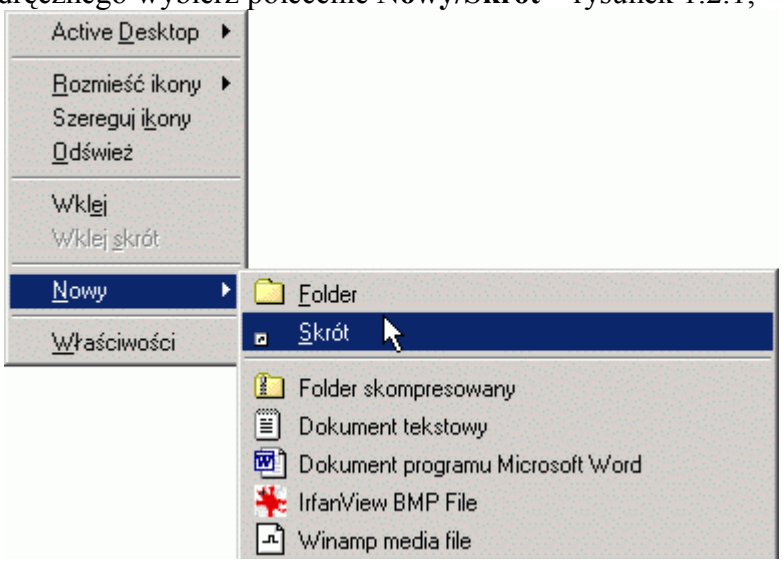
UWAGA: Wszystkie te polecenia można wykonać w systemie Windows, wykorzystując program Windows Commander.

Ćwiczenie 1.2. Tworzenie skrótu

Utwórz na Pulpicie skrót do programu Turbo Pascal.

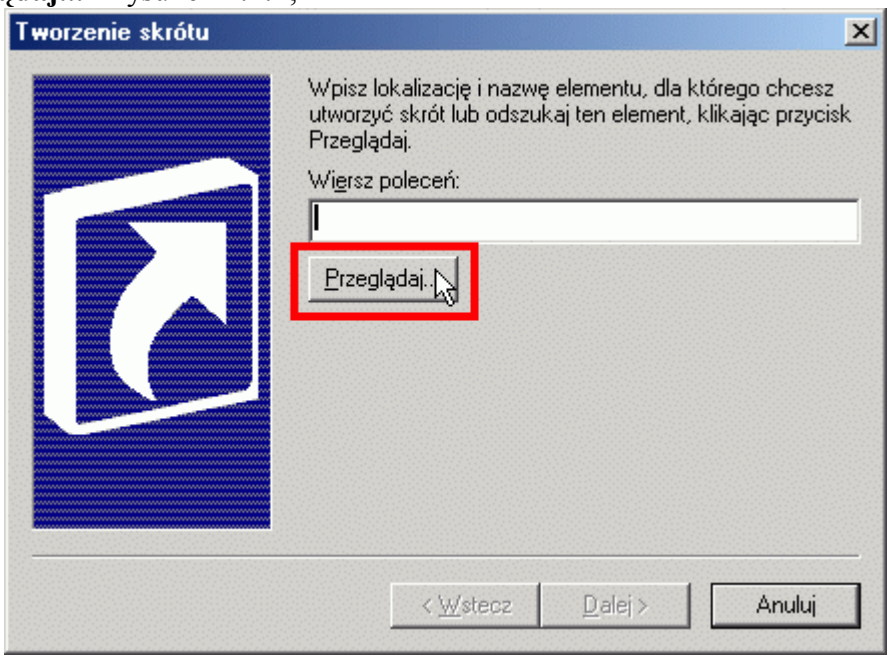
Sposób wykonania

- Kliknij prawym klawiszem myszy na wolnym obszarze PULPITU;
- Z menu podręcznego wybierz polecenie **Nowy/Skrót** – rysunek 1.2.1;



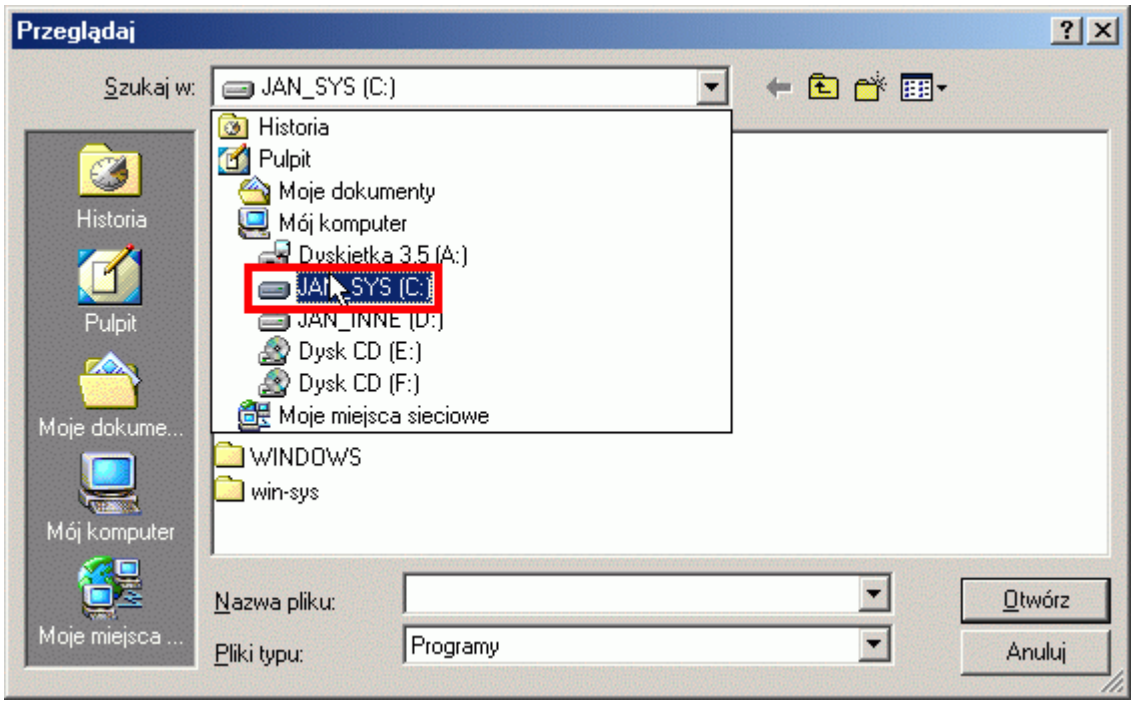
Rysunek 1.2.1. Widok wybranego polecenia Nowy/Skrót

- Będąc w oknie **Tworzenie skrótu**, kliknij lewym klawiszem myszy na przycisk **Przeglądaj...** – rysunek 1.2.2;



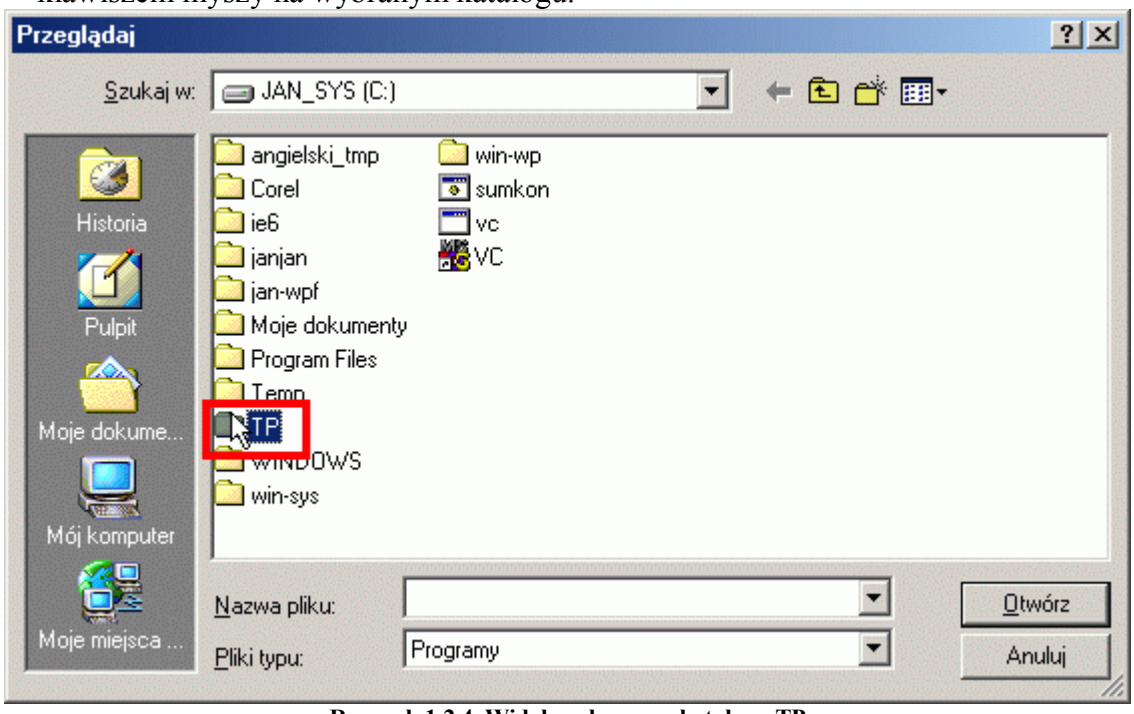
Rysunek 1.2.2. Widok wskazanego przycisku Przeglądaj

- Będąc w oknie **Przeglądaj**, wybierz z listy rozwijanej **Szukaj w** dysk twardy „C:” – rysunek 1.2.3;



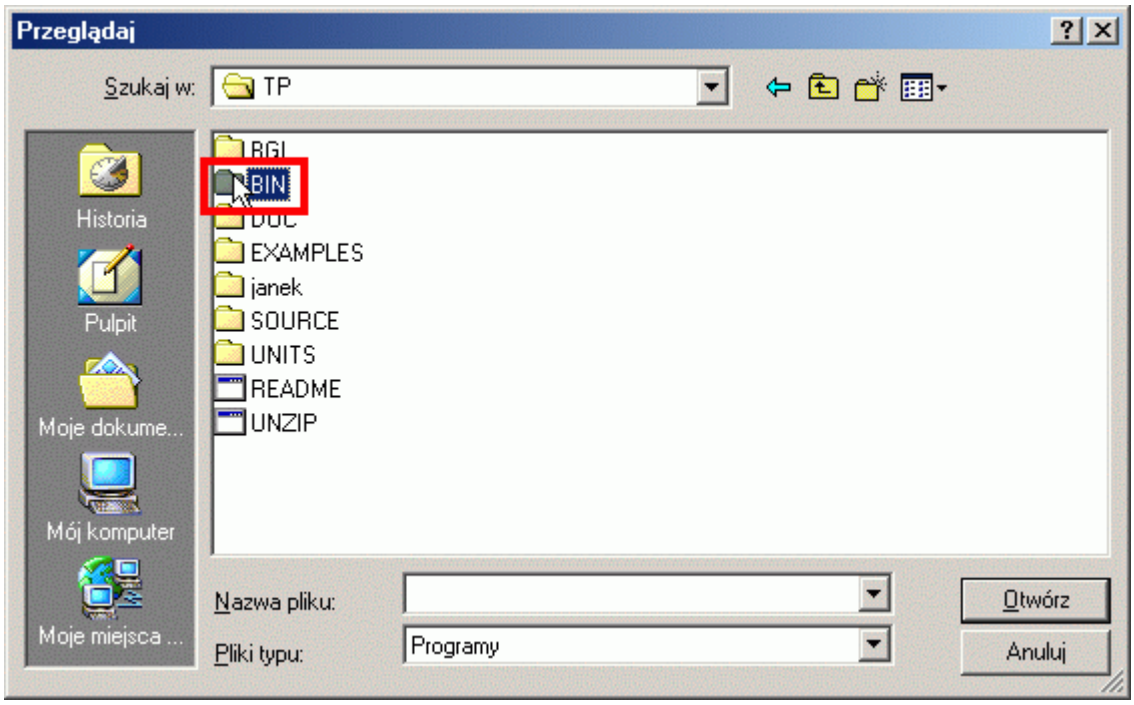
Rysunek 1.2.3. Widok wybranego dysku „C:" z listy rozwijanej „Szukaj w"

- Będąc w katalogu głównym dysku twardego „C:", wybierz katalog **TP** – rysunek 1.2.4;
  UWAGA: wyboru katalogu, dokonujemy przez dwukrotne kliknięcie lewym klawiszem myszy na wybranym katalogu.



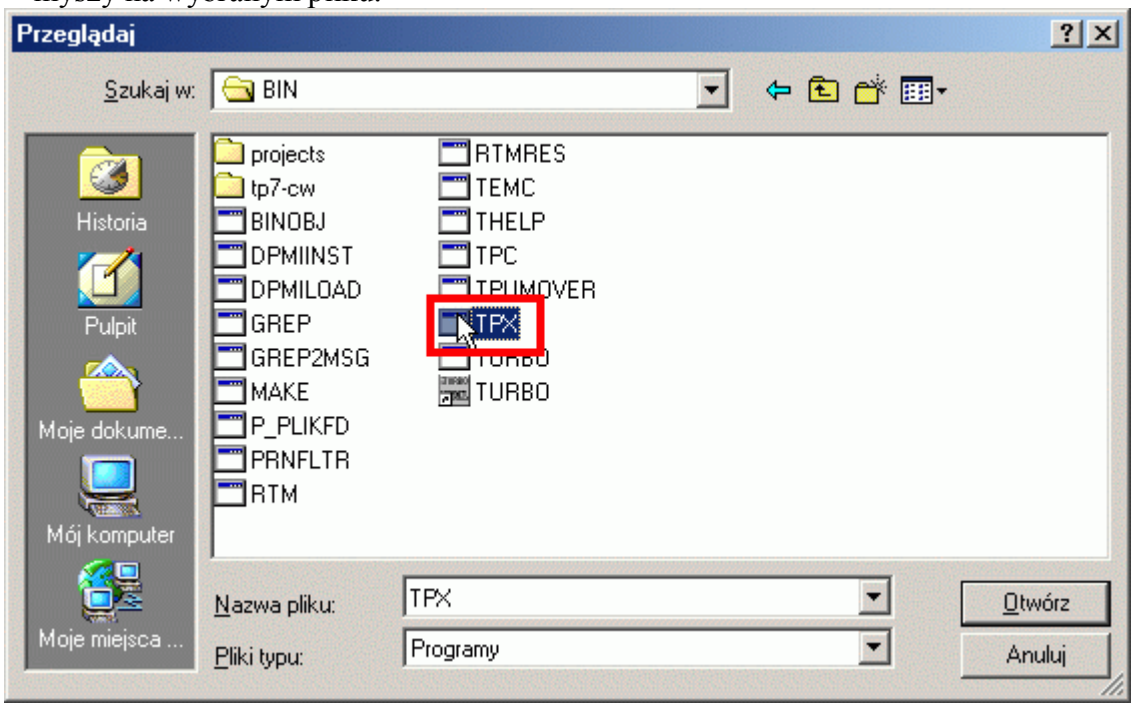
Rysunek 1.2.4. Widok wybranego katalogu TP

- Będąc w katalogu **TP**, wybierz katalog **BIN** – rysunek 1.2.5;



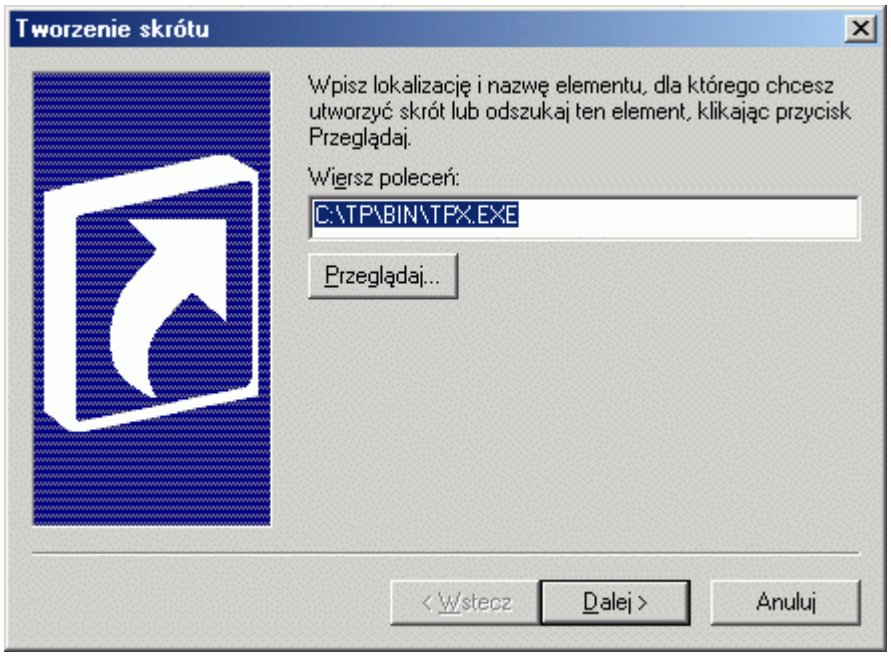
Rysunek 1.2.5. Widok wybranego katalogu BIN

- Będąc w katalogu **BIN**, wybierz plik **TPX.EXE** lub **TURBO.EXE** – rysunek 1.2.6; **UWAGA**: wyboru pliku, dokonujemy przez dwukrotne kliknięcie lewym klawiszem myszy na wybranym pliku.



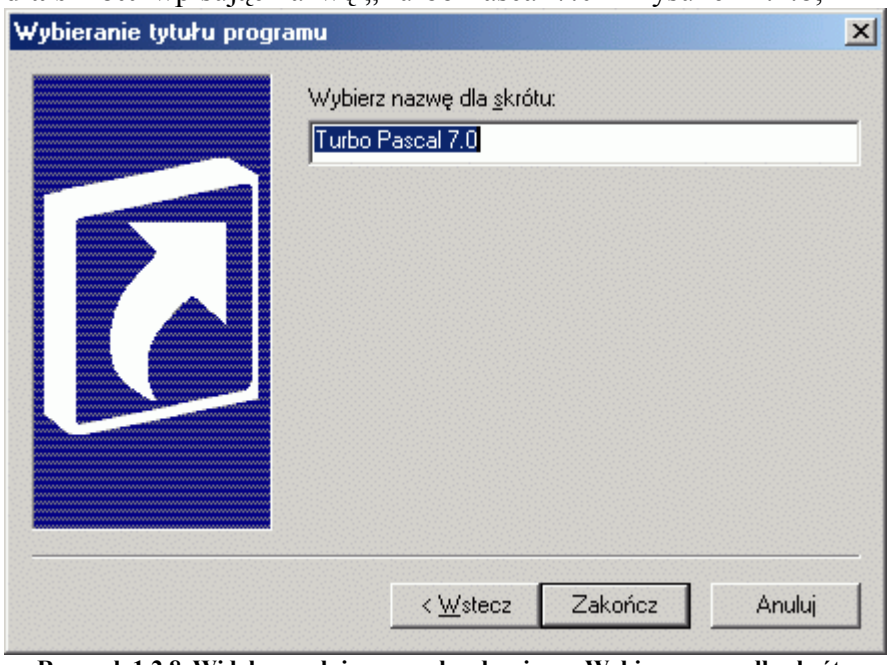
Rysunek 1.2.6. Widok wybranego pliku TPX.EXE

- Po wybraniu pliku, jego nazwa zostanie wprowadzona do pola edycyjnego **Wiersz poleceń** – rysunek 1.2.7;



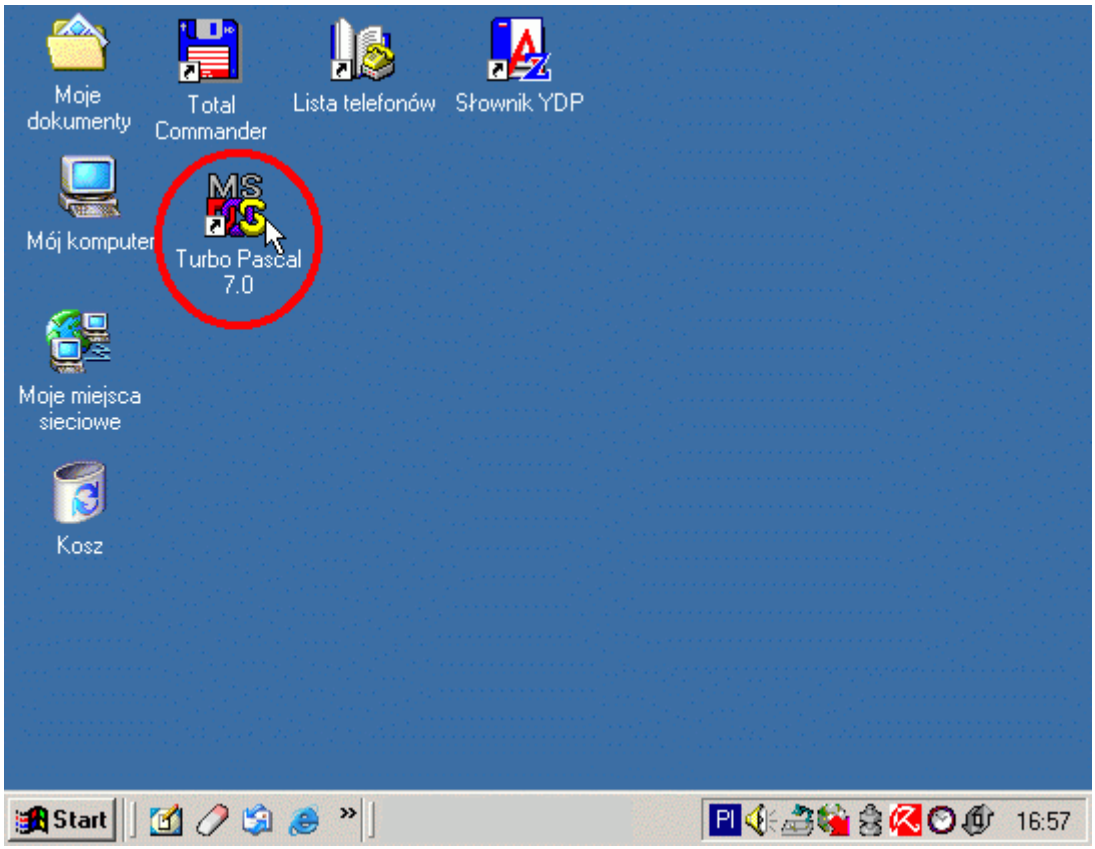
Rysunek 1.2.7. Widok wypełnionego pola edycyjnego Wiersz poleceń

- Następnie kliknij lewym klawiszem myszy na przycisk **Dalej**;
- Będąc w oknie **Wybieranie tytułu programu**, wypełnij pole edycyjne **Wybierz nazwę dla skrótu** wpisując nazwę „Turbo Pascal 7.0" – rysunek 1.2.8;



Rysunek 1.2.8. Widok wypełnionego pola edycyjnego Wybierz nazwę dla skrótu

- Kliknij lewym klawiszem myszy na przycisk **Zakończ**, co spowoduje utworzenie skrótu do programu **Turbo Pascal 7.0** na pulpicie – rysunek 1.2.9;



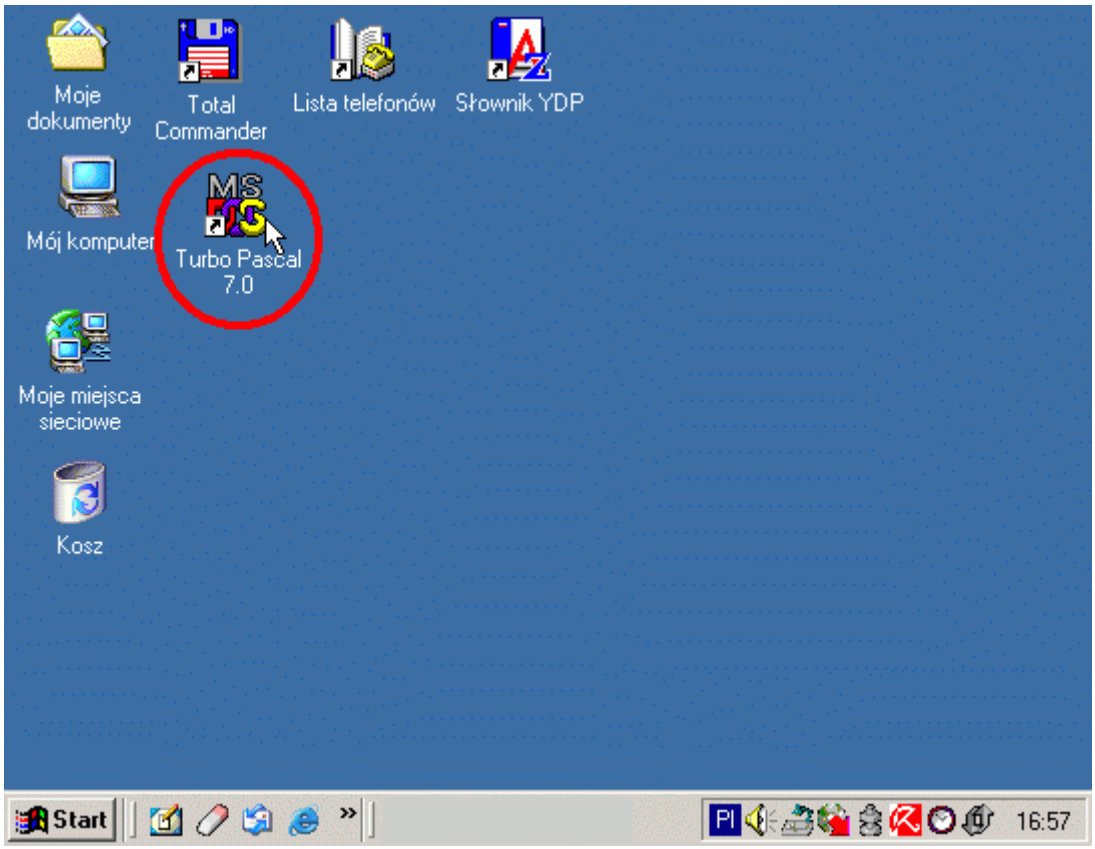
Rysunek 1.2.9. Widok utworzonego skrótu do programu Turbo Pascal 7.0

Ćwiczenie 1.3. Uruchomienie programu Turbo Pascal w systemie Windows

Uruchom program Turbo Pascal w systemie Windows.

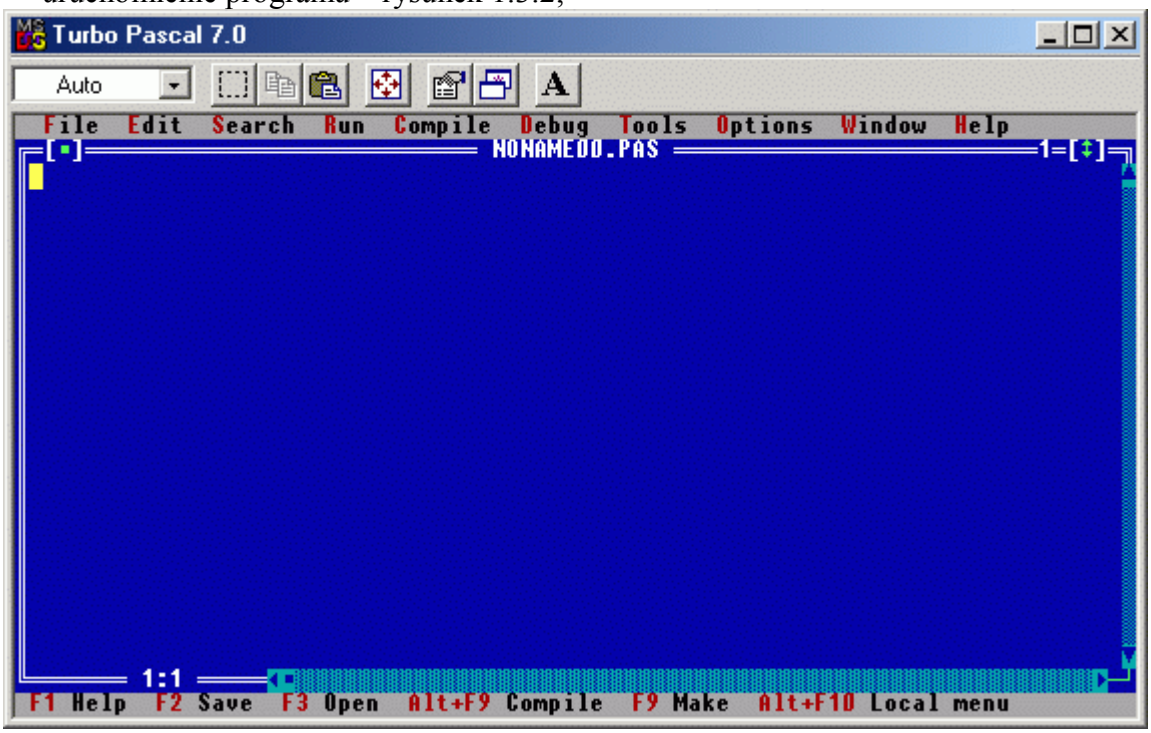
Sposób wykonania

- Wskaż kursorem myszy znajdujący się na pulpicie skrót, który reprezentuje program **Turbo Pascal 7.0** – rysunek 1.3.1;



Rysunek 1.3.1. Widok wybranego skrótu do programu Turbo Pascal

- Kliknij dwukrotnie lewym klawiszem myszy na tym skrócie, co spowoduje uruchomienie programu – rysunek 1.3.2;



Rysunek 1.3.2. Widok uruchomionego programu Turbo Pascal w systemie Windows

Od tego momentu możesz pracować w zintegrowanym środowisku programowania **Borland Turbo Pascal**.

2. Środowisko programistyczne

W rozdziale tym przedstawiono ćwiczenia, mające na celu zapoznanie czytelnika z podstawami obsługi zintegrowanego środowiska programistycznego **Borland Turbo Pascal 7.0**.

Ćwiczenie 2.1. Utworzenie nowego dokumentu

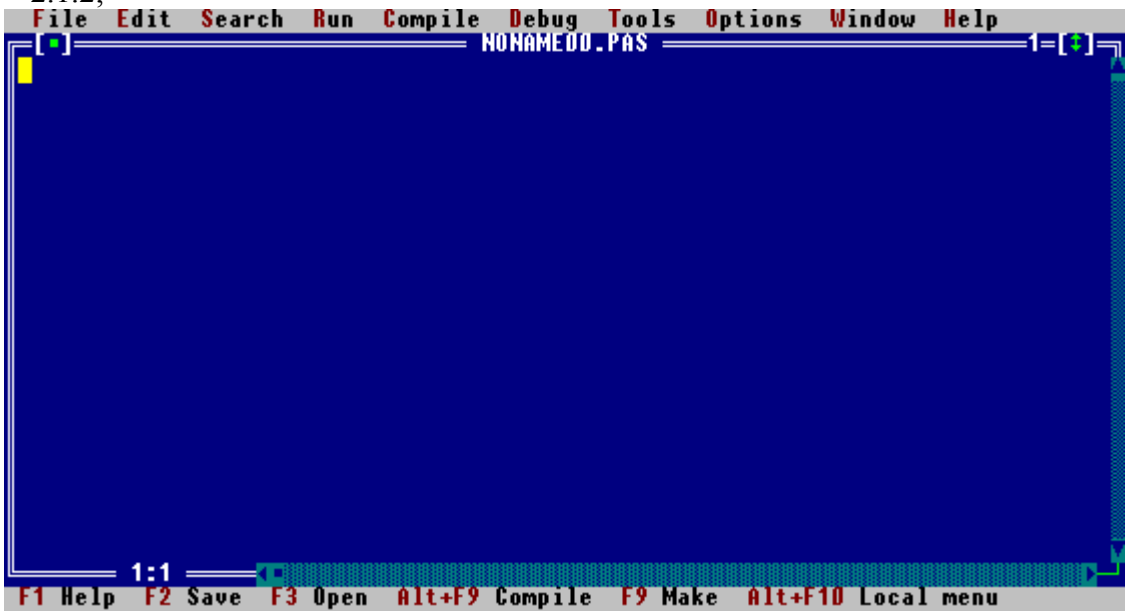
Utwórz nowy pusty dokument.

Sposób wykonania

- Wybierz polecenie **File/New** (Plik/Nowy) z menu górnego – rysunek 2.1.1;

Rysunek 2.1.1. Widok wybranego polecenia New (Nowy) z menu File (Plik)

- Po wybraniu tego polecenia zostanie utworzony nowy pusty dokument – rysunek 2.1.2;



Rysunek 2.1.2. Widok pustego dokumentu

Ćwiczenie 2.2. Otwarcie pliku z kodem źródłowym

Otwórz dowolny plik z kodem źródłowym (np. cw01.pas).

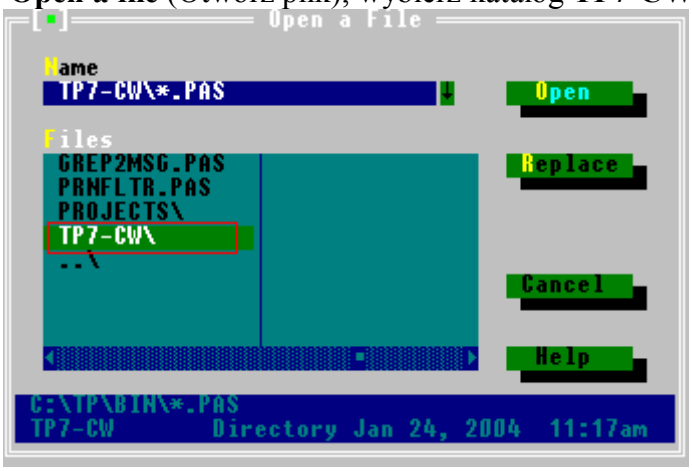
Opis:
Kod źródłowy to zrozumiały dla programisty tekst programu, napisany w dowolnym języku programowania (np. Turbo Pascal, C++, Delphi, Builder, itp.).

Sposób wykonania

- Naciśnij na klawisz funkcyjny **F3** lub wybierz polecenie **File/Open** (Plik/Otwórz) z menu górnego – rysunek 2.2.1;

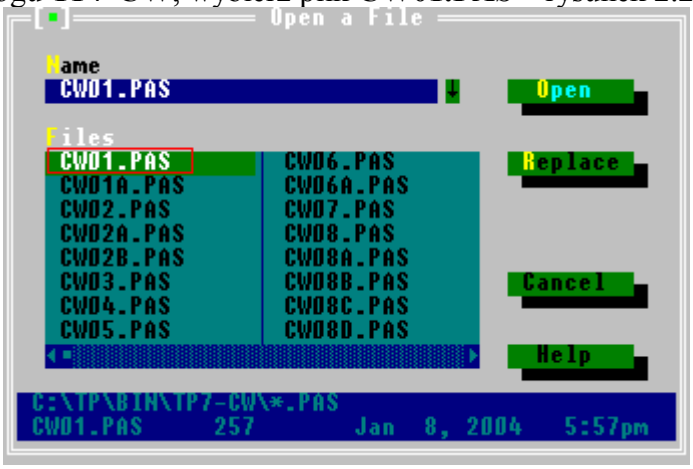
Rysunek 2.2.1. Widok wybranego polecenia Open (Otwórz)

- Będąc w oknie **Open a file** (Otwórz plik), wybierz katalog **TP7-CW** – rysunek 2.2.2;



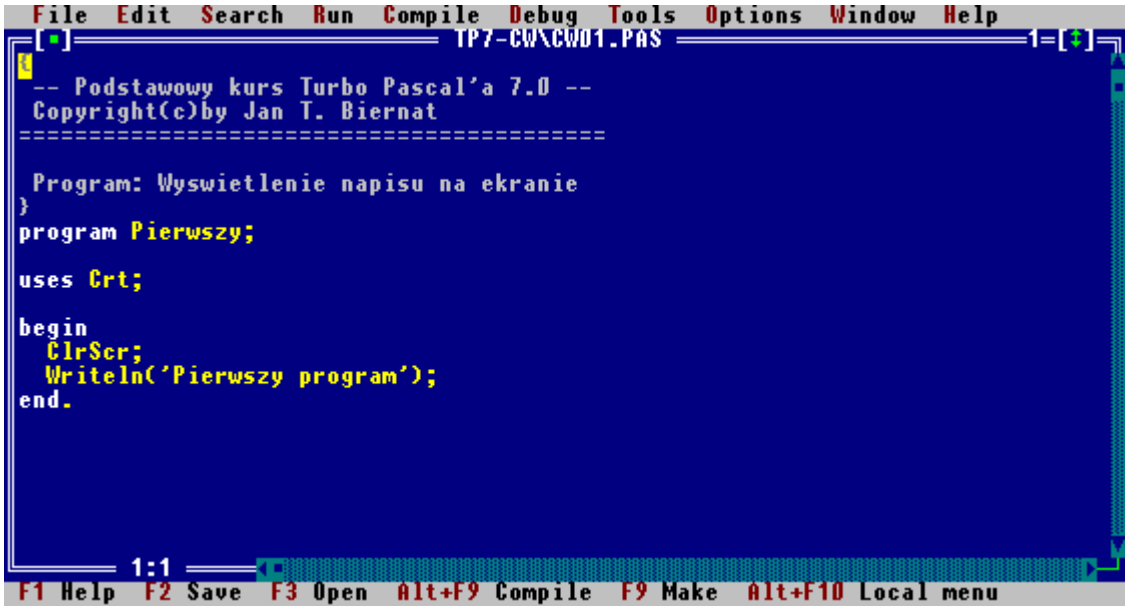
Rysunek 2.2.2. Widok wybranego katalogu TP7-CW

- Będąc w katalogu **TP7-CW**, wybierz plik **CW01.PAS** – rysunek 2.2.3;



Rysunek 2.2.3. Widok wybranego pliku CW01.PAS

- Naciśnij klawisz ENTER, co spowoduje wczytanie pliku do edytora – rysunek 2.2.4;



Rysunek 2.2.4. Widok wczytanego kodu źródłowego pliku CW01.PAS

Ćwiczenie 2.3. Zapisanie programu do pliku

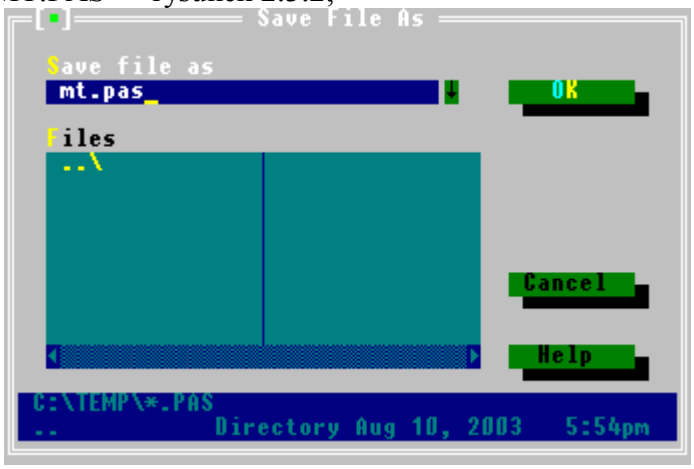
Zapisz program do pliku o nazwie np. MT.PAS.

Sposób wykonania

- Naciśnij klawisz funkcyjny **F2** lub wybierz polecenie **File/Save** (Plik/Zapisz) z menu górnego – 2.3.1;

Rysunek 2.3.1. Widok wybranego polecenia Save (Zapisz)

- Będąc w oknie **Save File As** (Zapisz plik jako), wpisz w polu edycyjnym **Save file as** nazwę pliku „MT.PAS" – rysunek 2.3.2;



Rysunek 2.3.2. Widok wypełnionego pola edycyjnego Save file as (Zapisz plik jako)

- Następnie naciśnij klawisz ENTER, co spowoduje zapisanie pliku pod wcześniej wpisaną nazwą.

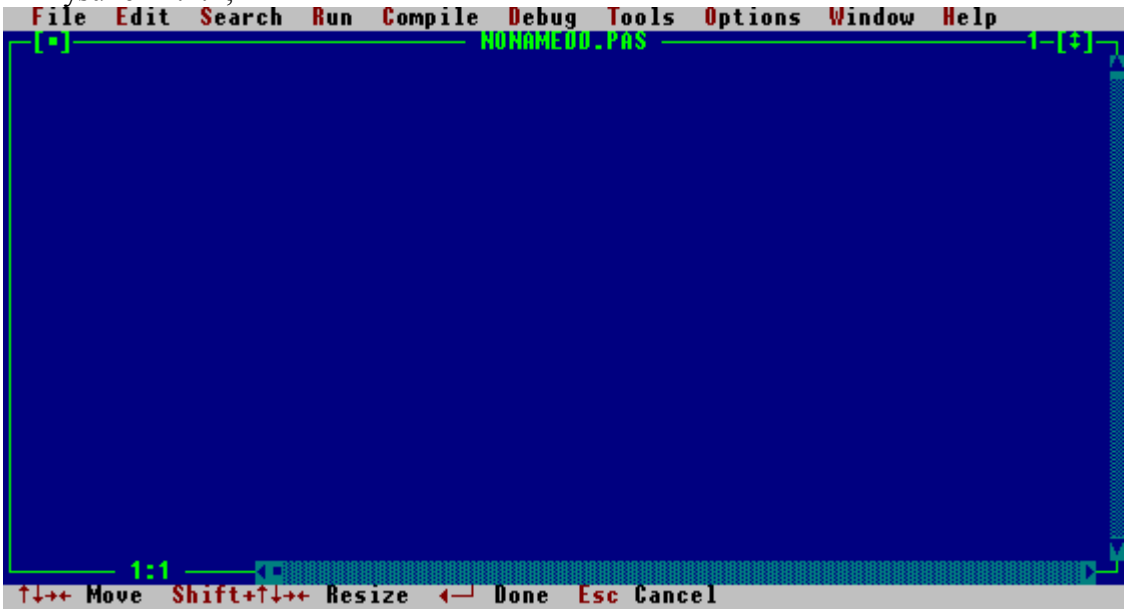
Ćwiczenie 2.4. Praca z oknami

Zaprezentuj skróty klawiszowe pomocne przy pracy z oknami w zintegrowanym środowisku programistycznym Turbo Pascal.

Sposób wykonania

Do pracy z oknami wykorzystywane są następujące skróty klawiszowe:

- **ESC**, wyjście z dowolnego okna lub aktualnie działającej funkcji/polecenia;
- **CTRL+F5**, umożliwia przestawienie okna w tryb przesuwania go po ekranie – rysunek 2.4.1;



Rysunek 2.4.1. Widok okna po przełączeniu go w tryb przesuwania po ekranie

Po włączeniu trybu przesuwania okien po ekranie, możesz wykorzystać następujące skróty klawiszowe:

- ❖ Klawisze nawigacyjne (tj. ←↑↓→), umożliwiają przesuwanie aktywnego okna po ekranie;
- ❖ SHIFT+←↑↓→, umożliwiają zmianę rozmiarów aktywnego okna;

- **ALT+CYFRA**, umożliwia przełączanie się pomiędzy oknami;
- **ALT+0**, wywołuje okno **Windows list** (Lista okien), która również umożliwia przełączanie się pomiędzy oknami oraz zamykanie okien;
- **F6**, umożliwia przełączanie się pomiędzy oknami w przód;
- **SHIFT+F6**, umożliwia przełączanie się pomiędzy oknami w tył;
- **F5**, umożliwia powiększenie/zmniejszenia aktywnego okna;
- **ALT+F3**, umożliwia zamknięcie aktywnego okna;

Ćwiczenie 2.5. Praca z kodem programu

Zaprezentuj skróty klawiszowe pomocne przy pisaniu programu w zintegrowanym środowisku programistycznym Turbo Pascal.

Sposób wykonania

Przy pisaniu programu pomocne są następujące skróty klawiszowe:

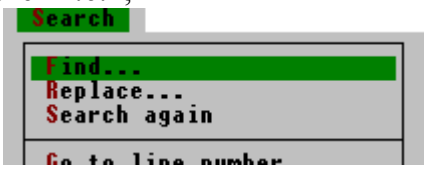
- **CTRL+Q** i **F**, wywołuje okno umożliwiające wyszukania fragmentu tekstu;
- **CTRL+L**, kontynuacja szukanego fragmentu tekstu;
- **CTRL+Q** i **A**, wywołuje okno umożliwiające zamianę fragmentu tekstu na inny fragment tekstu;
- **SHIFT+**←↑↓→, umożliwiają zaznaczanie fragmentu tekstu;
- **CTRL+INS**, umożliwia skopiowanie zaznaczonego fragmentu tekstu;
- **SHIFT+INS**, umożliwia wklejenie wcześniej skopiowanego fragmentu tekstu;
- **SHIFT+DEL**, umożliwia usunięcie zaznaczonego fragmentu tekstu;

Ćwiczenie 2.6. Szukanie tekstu

Znajdź w kodzie źródłowym wyraz „BEGIN”.

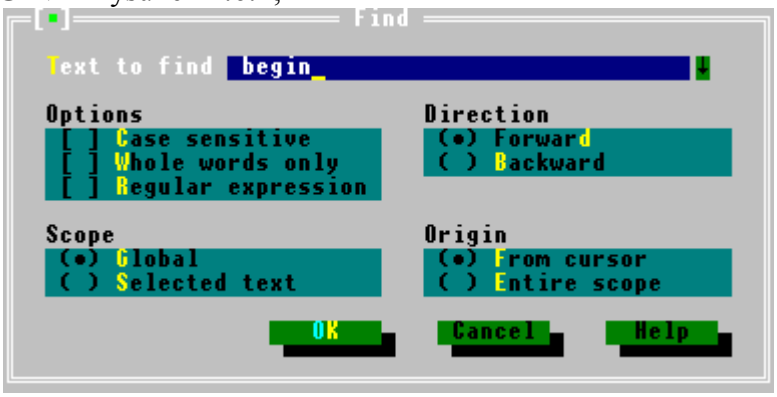
Sposób wykonania

- Naciśnij kombinację klawiszy **CTRL+Q** i **F** lub wybierz polecenie **Search/Find…** (Szukaj/Znajdź…) – rysunek 2.6.1;



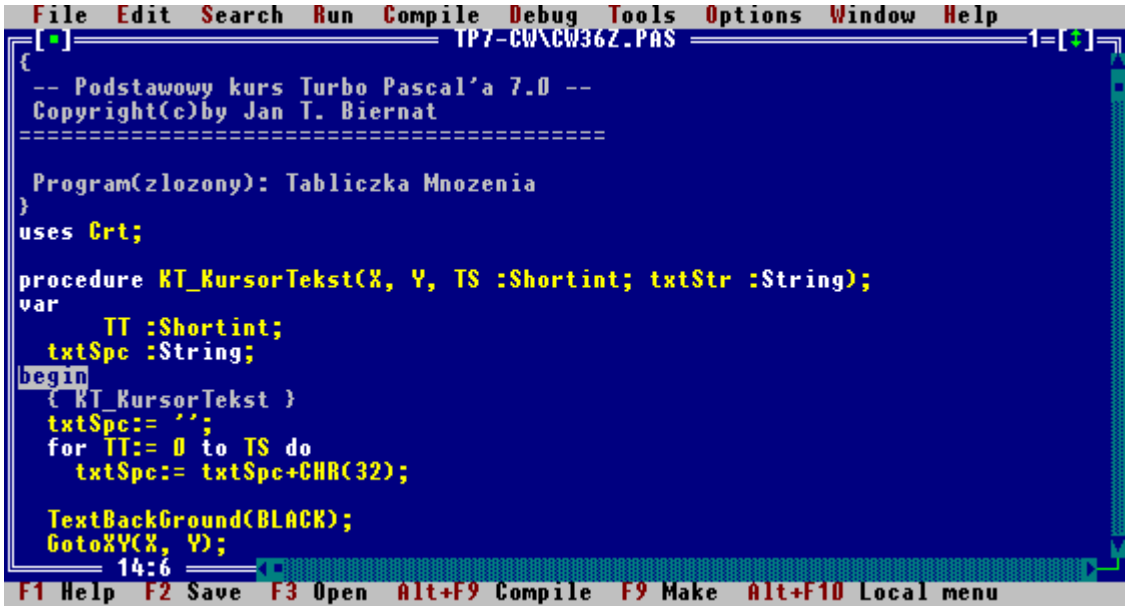
Rysunek 2.6.1. Widok wybranego polecenia Find(Znajdź)

- Będąc w oknie **Find** (Znajdź), wpisz w polu edycyjnym **Text to find** (Szukany tekst) wyraz „BEGIN” – rysunek 2.6.2;



Rysunek 2.6.2. Widok wpisanego wyrazu „BEGIN” w polu edycyjnym Text to find (Szukany tekst)

- Naciśnij klawisz ENTER;
- Gdy szukany ciąg znaków istnieje w tekście, to po odnalezieniu zostanie podświetlony – rysunek 2.6.3;



Rysunek 2.6.3. Widok znalezionego wyrazu „BEGIN"

UWAGA: W przypadku, gdy wyraz nie zostanie odnaleziony, to zostanie wyświetlony komunikat **Search string not found** (Szukany ciąg znaków nie został odnaleziony) – rysunek 2.6.4.

Rysunek 2.6.4. Widok komunikatu informującego o braku szukanego ciągu znaków

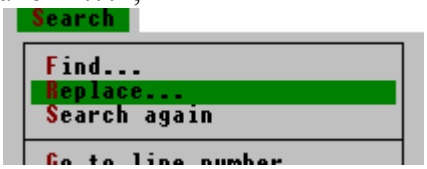
W celu kontynuacji szukanego wyrazu „BEGIN", wystarczy nacisnąć kombinację klawiszy **CTRL+L**.

Ćwiczenie 2.7. Zamiana tekstu

Zamień wyraz „BEGIN" na wyraz „START".

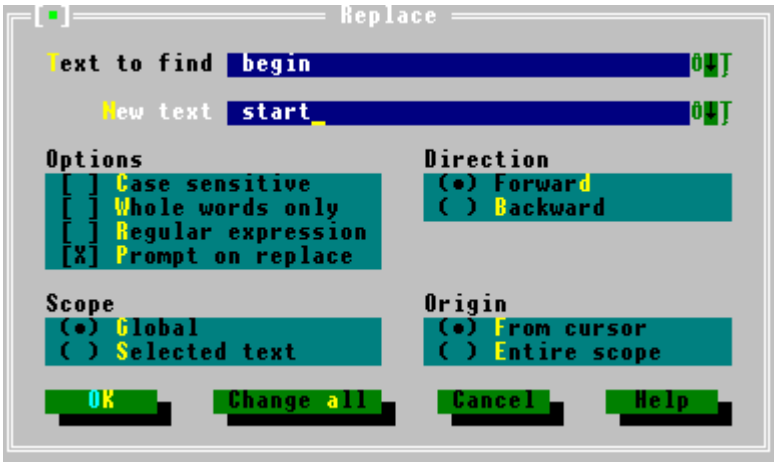
Sposób wykonania

- Naciśnij kombinację klawiszy **CTRL+Q** i **A** lub wybierz polecenie **Search/Replace…** (Szukaj/Zamień…) – rysunek 2.7.1;



Rysunek 2.7.1. Widok wybranego polecenia Replace (Zamień)

- Będąc w oknie **Replace** (Zamień), w polu edycyjnym **Text to find** (Szukany tekst) wpisz wyraz „BEGIN", natomiast w polu edycyjnym **New text** (Nowy tekst) wpisz wyraz „START" – rysunek 2.7.2;



Rysunek 2.7.2. Widok wypełnionych pól edycyjnych w oknie Replace (Zamień)

- Naciśnij kombinację klawiszy **ALT+A**;
- Gdy zostanie odnaleziony wyraz „BEGIN”, to zostanie wyświetlone okno **Information** (Informacja) z pytaniem **Replace this occurrence?** (Zamienić znaleziony wyraz) – rysunek 2.7.3;

Rysunek 2.7.3. Widok okna pytającego o potwierdzenie zamiany znalezionego wyrazu

UWAGA: Naciśnięcie klawisza **Y** lub ENTER spowoduje zamianę znalezionego wyrazu. Natomiast naciśnięcie klawisza **N**, spowoduje szukanie następnego wystąpienia szukanego wyrazu.

UWAGA: W przypadku, gdy wyraz nie zostanie odnaleziony, to zostanie wyświetlony komunikat **Search string not found** (Szukany ciąg znaków nie został odnaleziony) – rysunek 2.7.4.

Rysunek 2.7.4. Widok komunikatu informującego o braku szukanego ciągu znaków

W celu kontynuacji szukanego wyrazu „BEGIN”, wystarczy nacisnąć kombinację klawiszy **CTRL+L**.

Ćwiczenie 2.8. Uruchomienie programu

Uruchom program o nazwie „cw36z.pas”.

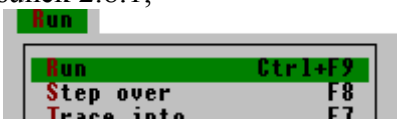
Opis:
Kod wynikowy powstaje w wyniku kompilacji (tj. tłumaczenia) kodu źródłowego (napisanego w dowolnym języku programowania) na kod maszynowy zrozumiały dla procesora.

Kompilację kodu źródłowego wykonują programy zwane kompilatorami. Proces kompilacji polega na analizie kodu źródłowego i dopiero po bezbłędnej analizie następuje generowanie kodu maszynowego. W przypadku wykrycia błędu, po poprawienie go kompilacja jest wykonywana od początku. Program otrzymany po kompilacji może być uruchomiony poza środowiskiem programistycznym, w którym został stworzony. Kod wynikowy zależy od rodzaju procesora, tak więc dla każdego procesora trzeba wykonać osobną kompilację wykorzystując odpowiedni program kompilujący. Typowym przykładem kompilatora jest Turbo Pascal, C++, Delphi, Builder.

Interpreter to program, który zamienia kod źródłowy (napisany w dowolnym języku programowania) na kod maszynowy zrozumiały dla procesora. Interpreter analizuje kod źródłowy linia po linii wykonując przy każdej linii tłumaczenie na kod maszynowy. W przypadku napotkania błędu interpreter wstrzymuje tłumaczenie. Po poprawieniu błędu, tłumaczenie rozpoczyna się od linii, w której on wystąpił. Kod wygenerowany przy pomocy interpreterów jest duży i mniej efektywny. Typowym przykładem interpretera jest język BASIC.

Sposób wykonania

- Wczytaj program – patrz ćwiczenie 2.2;
- Naciśnij kombinację klawiszy **CTRL+F9** lub wybierz polecenie **Run/Run** (Uruchom/Uruchom) – rysunek 2.8.1;



Rysunek 2.8.1. Widok wybranego polecenia Run (Uruchom)

- Widok działającego programu przedstawia rysunek 2.8.2;

Rysunek 2.8.2. Widok programu Tabliczka mnożenia

Ćwiczenie 2.9. Korzystanie z pomocy

Zaprezentuj sposoby korzystania z pomocy.

Sposób wykonania

Sposób 1

- Naciśnij kombinację klawiszy **SHIFT+F1** lub wybierz menu **Help/Index** (Pomoc/Indeks) – rysunek 2.9.1;

Rysunek 2.9.1. Widok wybranego polecenia Help/Index (Pomoc/Indeks)

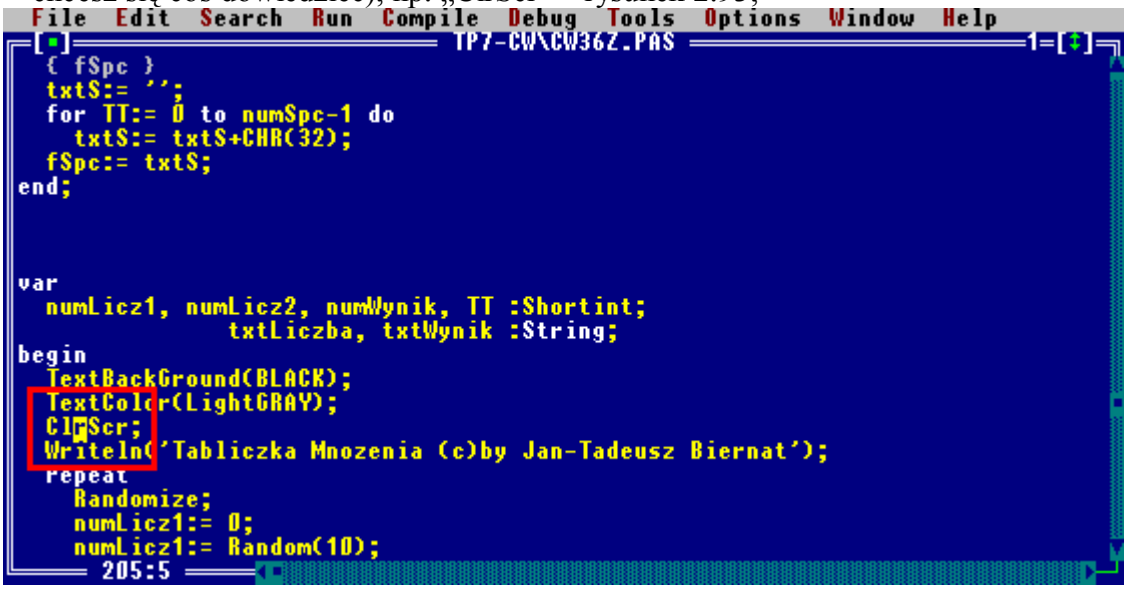
- Będąc w oknie **Help** (Pomoc), zacznij wypisywać na klawiaturze instrukcje „ClrScr”, co spowoduje jej wyszukanie – rysunek 2.9.2;

Rysunek 2.9.2. Widok znalezionej instrukcji „ClrScr”

- Po znalezieniu instrukcji naciśnij klawisz ENTER, co spowoduje wyświetlenie informacji na temat instrukcji „ClrScr” – rysunek 2.9.4;

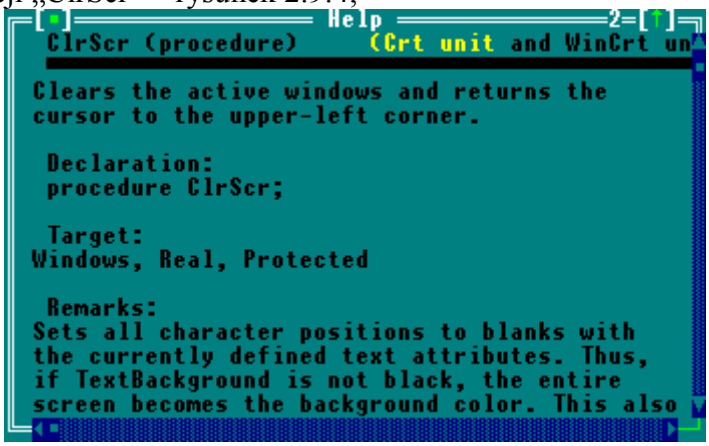
Sposób 2

- Będąc w kodzie programu, najedź kursorem tekstowym na instrukcje (na temat której chcesz się coś dowiedzieć), np. „ClrScr” – rysunek 2.93;



Rysunek 2.9.3. Widok instrukcji „ClrScr" wskazanej kursorem tekstowym

- Naciśnij kombinację klawiszy **CTRL+F1**, co spowoduje wyświetlenie informacji na temat instrukcji „ClrScr" – rysunek 2.9.4;



Rysunek 2.9.4. Widok informacji na temat instrukcji „ClrScr"

3. Podstawy Pascal'a

W rozdziale tym przedstawiono ćwiczenia, mające na celu zapoznanie czytelnika z podstawowymi konstrukcjami języka Borland Turbo Pascal 7.0.

Ćwiczenie 3.1. Komentarze

Zaprezentuj możliwość opisania programu.

Sposób wykonania

Komentarze wykorzystywane są do opisania czynności wykonywanych w określonym fragmencie kodu oraz czynności wykonywanych przez poszczególne instrukcje. Komentarze mogą być również wykorzystywane do opisania zmiennych i stałych.
Mają one na celu zwiększyć czytelność programu, a co za tym idzie ułatwić późniejsze modyfikacje dokonywane przez programistę.

Poniżej zamieszczone są przykłady komentarzy:

- { } - Pomiędzy nawiasami klamrowymi możesz dać komentarz, np. *{ to jest komentarz }*;
- (* *) – Pomiędzy tymi znakami również możesz umieścić komentarz, np. *(* To jest również komentarz *)*;

Ćwiczenie 3.2. Typy danych

Omów rodzaje typów danych.

Sposób wykonania

Typ jest zbiorem określającym wartości, jakie może przyjmować zmienna lub stała. Rozróżnia się wymienione niżej typy.

- Typ całkowity

Nazwa	Zakres	Rozmiar
Shortint	–128 … 127	8-bitów (1 bajt)
Integer	–32768 … 32767	16-bitów (2 bajty)
Longint	–2147483648 … 2147483647	32-bitów (4 bajty)
Byte	0 … 255	8-bitów (1 bajt)
Word	0 … 65535	16-bitów (2 bajty)

- Typ rzeczywisty

Nazwa	Zakres	Ilość cyfr	Rozmiar
Real	$2.9x10^{-39}..1.7x10^{38}$	11-12	6 bajtów
Single	$1.5x10^{-45}..3.4x10^{38}$	7-8	4 bajtów
Double	$5.0x10^{-324}..1.7x10^{308}$	15-16	8 bajtów
Extended	$3.4x10^{-4932}..1.1x10^{4932}$	19-20	10 bajtów
Comp	$-9.2x10^{18}..9.2x10^{18}$	19-20	8 bajtów

- Typ logiczny

Najczęściej używanym typem logicznym jest typ BOOLEAN, który może przyjmować dwa stany. Pierwszy stan to TRUE (Prawda – logiczna 1), natomiast drugi stan to FALSE (Fałsz – logiczne 0).

- Typ znakowy
  - String – może przechowywać dowolne znaki, a ilość tych znaków musi mieścić się w zakresie od 1 do 255;
  - Char – może przechowywać dowolny jeden znak;

Ćwiczenie 3.3. Pierwszy program

Napisz program, który wyświetli tekst „Pierwszy program”.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW01.PAS**.

```
program Pierwszy;
{
Nagłówek programu składa się z następujących elementów:
   • Słowa kluczowego PROGRAM;
   • Nazwy (identyfikatora) programu występującego po słowie kluczowym PROGRAM. Tą
     nazwą (identyfikatorem) jest wyraz np. „Pierwszy”.
}


uses Crt;
{
 Po słowie USES następuje deklaracja modułu (lub modułów), którego instrukcje są
wykorzystywane w aktualnie pisanym programie.
Moduł CRT jest odpowiedzialny za obsługę ekranu i klawiatury.
}

begin
  ClrScr; { Czyści zawartość ekranu }

 { Wyświetla na ekran tekst znajdujący się pomiędzy apostrofami }
  Writeln('Pierwszy program');
end.
```

Naciśnij kombinację klawiszy CTRL+F9, w celu uruchomienia programu.

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.4. Wizytówka

Napisz program, który wyświetli informację o autorze programu.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW01A.PAS**.

```
uses Crt;

begin
  ClrScr; { Czyści zawartość ekranu }
  Writeln; { Przejście do następnego wiersza }


  Writeln(' ************************************');
  Writeln(' * Podstawowy kurs                 *');
  Writeln(' * Turbo Pascal`a 7.0              *');
  Writeln(' * Copyright(c)by Jan-Tadeusz Biernat *');
  Writeln(' ************************************');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.5. Oczekiwanie na naciśnięcie dowolnego klawisza

Napisz program, który wymusi na użytkowniku naciśnięcie klawisza po wyświetleniu dowolnego tekstu.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW02.PAS**.

```
uses Crt;

begin
  ClrScr; { Czyści zawartość ekranu }

{ Wyświetla na ekran tekst znajdujący się pomiędzy apostrofami, tj. znakami „ ’ ”}
  Writeln('== Turbo Pascal 7.0 ==');
  Writeln; { Przejście do następnego wiersza }

  Write('Naciśnij dowolny klawisz...');
  repeat
  until KeyPressed;
  {
  Funkcja „KeyPressed” służy do sprawdzenia, czy nastąpiło naciśnięcie na klawiaturze
dowolnego klawisza. Jeżeli klawisz został naciśnięty, to funkcja „KeyPressed” przyjmuje
wartość TRUE. W przeciwnym przypadku wartością funkcji jest FALSE.

Pętla REPEAT…UNTIL będzie wykonywana, dopóki nie zostanie naciśnięty dowolny klawisz
na klawiaturze. Sprawdzenie, czy klawisz na klawiaturze został naciśnięty jest wykonywane na
końcu pętli REPEAT…UNTIL.
  }
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.6. Losowo wybierany kolor

Napisz program, który wyświetli tekst „Turbo Pascal 7.0" i będzie zmieniał kolor tekstu, do momentu naciśnięcia przez użytkownika dowolnego klawisza.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW02A.PAS**.

```
uses Crt;

begin
  ClrScr; { Czyści zawartość ekranu }
  Writeln; { Przejście do następnego wiersza }

{ Wyświetla na ekran tekst znajdujący się pomiędzy apostrofem, tj. znakiem „ ' "}
  Writeln('== Napis zmienia kolory ==');
  Writeln;
  repeat
    Randomize; { Zainicjowanie wbudowanego generatora liczb losowych }
    GotoXY(4, 4);
{
GotoXY(X, Y) – umieszcza tekst lub kursor w miejscu o współrzędnych X, Y. W tym przypadku
tekst umieszczony jest we współrzędnych 4, 4.
}

    TextColor(Random(9));
    {
     Random(LICZBA) – Losuje liczby z zakresu podanego w parametrze LICZBA (tj. z zakresu
od 0 do 9).
    TextColor(KOLOR) – umożliwia nadanie znakom koloru, który trzeba podać w parametrze
KOLOR.
    }

    Writeln('Turbo Pascal 7.0');
  until KeyPressed;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.7. Wyświetlanie tekstu literka po literce

Napisz program, który wyświetli podany tekst literka po literce.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW02B.PAS**.

```
uses Crt;
```

```
const
{ Deklaracja stałej „txtNapis” przechowującej dowolny tekst }
  txtNapis = 'Wszystko powinno być tak proste, jak to tylko możliwe, ale nie prostsze. - Albert
Einstein';

var
  numRuch :Shortint; { Deklaracja zmiennej liczbowej całkowitej }
begin
  ClrScr; { Wyczyszczenie zawartości ekranu }
  Writeln; { Przejście do następnego wiersza }

{ Wyświetlenie tekstu znajdującego się pomiędzy znakami apostrofu, tj. znakami „ ‘ ” }
  Writeln('== Wyświetlenie tekstu znak po znaku ==');
  Writeln;

  numRuch:= 0; { Wyzerowanie zmiennej }
  repeat
    numRuch:= numRuch+1; { Zwiększenie zmiennej o wartość 1 }
    GotoXY(4, 4);
{
GotoXY(X, Y) – umieszcza tekst lub kursor w miejscu o współrzędnych X, Y. W tym przypadku
tekst umieszczony jest we współrzędnych 4, 4.
}

    Write(Copy(txtNapis, 1, numRuch));
    {
        Copy(Ciag_Znakow, Pozycja_Pierwszego_Znaku, Liczba_Znakow)

        Funkcja zwraca fragment ciągu znaków, o odpowiedniej ilości
        znaków, określonych w parametrze "Liczba_Znakow", od
        pozycji "Pozycja_Pierwszego_Znaku".
        UWAGA: Znak spacji traktowany jest jako znak.

        Na przykład: Copy('To jest tekst', 4, 6) - efektem będzie wyciągnięcie fragmentu tekstu
   "jest t".
    }

    Delay(222); { Zatrzymuje działanie programu na kilka milisekund, np. 222 }
  until KeyPressed or (numRuch >= Length(txtNapis));
  {
Pętla REPEAT…UNTIL w tym przykładzie służy do zwiększania wartości liczbowej,
przechowywanej w zmiennej „numRuch” o wartość 1. Dzięki temu wprowadzony tekst jest
wyświetlany litera po literze przez zwiększanie wartości zmiennej „numRuch”. Postawione
warunki na końcu pętli (tj. „KeyPressed” i „numRuch >= Length(txtNapis)”) powodują, że
literowanie będzie trwało, aż do wyświetlenia całego tekstu lub do momentu naciśnięcia
dowolnego klawisza.

Length(S) – funkcja ta zwraca liczbę znaków, które zawiera łańcuch S.
}
```

end.

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.8. Wprowadzanie danych z klawiatury

Napisz program, który umożliwi wyświetlenie wprowadzonych z klawiatury danych.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW03.PAS**.

```
uses Crt;

var
  D1, D2 :String; { Deklaracja zmiennych tekstowych }
begin
  ClrScr; { Czyści zawartość ekranu }

{ Wyświetla na ekran tekst znajdujący się pomiędzy apostrofami }
  Writeln('Program wyświetla wprowadzone dane przez użytkownika.');
  Writeln; { Przejście do następnego wiersza }
  Write('Podaj swoje imię:');

  Readln(D1);
 {
  Umożliwia wprowadzenie danych z klawiatury i przypisanie ich do zmiennej „D1”
  }

  Write('Podaj swoje nazwisko:');
  Readln(D2);
  Writeln;
  Writeln('Podales nastepujace dane:');

  Writeln('    Imię:'+CHR(32)+D1);
{ Wyświetlenie tekstu „Imię:” ze spacją oraz zawartością zmiennej „D1”.

 Znak spacji jest wyświetlany przy pomocy funkcji CHR(32).
Funkcja CHR(X) zwraca znak, którego kod ASCII określony jest w parametrze „X”, np.
CHR(65) wyświetli znak „A”.

ASCII – jest to tablica znaków, która opisuje jakie wartości liczbowe są przypisane
poszczególnym znakom. }

  Writeln('Nazwisko:'+CHR(32)+D2);
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.9. Funkcje

Omów strukturę funkcji i sposoby jej wywoływania.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW04.PAS**.

Funkcja składa się z kilku bloków.

- Definicja funkcji obejmuje
  - ✓ Nagłówek (nazwy) funkcji, który jest umieszczony zaraz po spacji, za słowem **function** (np. „**function** Sumuj”);
  - ✓ Parametry, które umieszczone są zaraz za nazwą funkcji, pomiędzy nawiasami (np. „**function** Sumuj(A, B :Integer)”);
  - ✓ Typ wyniku, jaki funkcja ma zwrócić – umieszczony jest on za nawiasem zamykającym (tj. „) ”), który poprzedza znak „:” (np. „**function** Sumuj(A, B :Integer): Integer”);
- Część opisową – deklaruje się tam stałe lokalne (po słowie **const**) lub zmienne lokalne (po słowie **var**);
- Część operacyjną, w której znajdują się instrukcje wykonujące określone czynności – instrukcje te są umieszczone pomiędzy słowem „**begin**”, a słowem „**end**;”.

Poniżej przedstawiono przykład definicji kilku funkcji.

```
uses Crt;

function JedenParametr(A :Integer) :Integer; { Definicja funkcji }
begin
  JedenParametr:= A*A;
 { Część operacyjna: Wykonuje mnożenie liczby przechowywanej w zmiennej „A” }
end;

function DwaParametry(A, B :Integer) :Integer; { Definicja funkcji }
begin
  DwaParametry:= A*B;
{ Część operacyjna: Wykonuje mnożenie liczb przechowywanych w zmiennych „A” i „B” }
end;

function DwaParametryRef(A :Integer; var B :Integer) :Integer;
{
 Parametr „B” funkcji „DwaParametryRef” poprzedzony jest słowem VAR, co umożliwia
dokonanie podmiany zawartości zmiennej „B”. Na przykład: A = 5, B = 10 i po wykonaniu
sumowania B = 15.
}
begin
  DwaParametryRef:= A*B;

  B:= A+B;
{ Część operacyjna: Dodawanie dwóch liczb przechowywanych w zmiennych „A” i „B” oraz
przypisanie zmiennej „B” wyniku dodawania }
```

```
end;


var
  Licz :Integer; { Deklaracja zmiennej liczbowej }
begin
  ClrScr; { Czyści zawartość ekranu }
  Writeln('Wprowadzono liczbę 4, wynik (*): ', JedenParametr(4));
{
 Wyświetlenie tekstu na ekranie oraz wywołanie funkcji „JedenParametr" z liczbą 4, jako
parametr.
}


  Writeln('Wprowadzono liczby 5 i 4, wynik (*): ', DwaParametry(5, 4));
{
 Wyświetlenie tekstu na ekranie oraz wywołanie funkcji „DwaParametry" z liczbami 5 i 4,
które traktowane są jako parametry.
}


  Licz:= 10; { Przypisanie zmiennej liczby 10 }
  Writeln('Wprowadzono liczby 5 i 10, wynik (*): ', DwaParametryRef(5, Licz));
{
 DwaParametryRef(5, Licz)
Wywołanie funkcji „DwaParametryRef" z dwoma parametrami.
Zwróć uwagę na to, że w drugim parametrze nie ma podanej liczby. Jest tak dlatego, że przed
parametrem „B" w definicji funkcji „DwaParametryRef" jest wyraz VAR. Mówi nam on o
tym, że przy wywołaniu funkcji w drugim parametrze należy umieścić zmienną tego samego
typu, co parametr znajdujący się w definicji funkcji. Zmienna ta umożliwia wprowadzenie (w
tym przykładzie) liczby 10, a po wykonaniu działania dodawania przechowa nam wynik (tj.
liczbę 15).
}

  Writeln('Wprowadzono liczby 5 i 10, wynik (+): ', Licz);
{
 Wyświetlenie tekstu, wraz z wynikiem, który jest przechowywany w zmiennej „Licz".
}
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.10. Funkcje doczytywane z innego pliku

Napisz program umożliwiający wykorzystanie funkcji z poprzedniego przykładu. Funkcje są zapisane w innym pliku niż sam program główny.

Sposób wykonania

Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW04A.PAS** i **CW04AA.PAS**.

Plik o nazwie **CW04AA.PAS** ma postać:

```
function JedenParametr(A :Integer) :Integer; { Definicja funkcji }
begin
  JedenParametr:= A*A;
{ Część operacyjna: Wykonuje mnożenie liczby przechowywanej w zmiennej „A” }
end;

function DwaParametry(A, B :Integer) :Integer; { Definicja funkcji }
begin
  DwaParametry:= A*B;
{ Część operacyjna: Wykonuje mnożenie liczb przechowywanych w zmiennych „A” i „B” }
end;
```

Plik o nazwie **CW04A.PAS** ma postać:

```
uses Crt;

{$I cw04aa.pas}
{ Powyższa dyrektywa umożliwia dołączenie pliku CW04AA.PAS do pliku CW04A.PAS. }


begin
  ClrScr; { Czyści zawartość ekranu }
  Writeln('Wprowadzono liczbę 4, wynik (*): ', JedenParametr(4));
{
 Wyświetlenie tekstu na ekranie oraz wywołanie funkcji „JedenParametr” z liczbą 4, jako
parametr.
}

  Writeln('Wprowadzono liczby 5 i 4, wynik (*): ', DwaParametry(5, 4));
{
 Wyświetlenie tekstu na ekranie oraz wywołanie funkcji „DwaParametry” z liczbami 5 i 4,
które są traktowane jako parametr.
}
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.11. Procedury

Omów strukturę procedury i sposoby jej wywoływania.

Sposób wykonania

Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW05.PAS**.

Procedura składa się z kilku bloków.

- Definicja procedury obejmuje
  - ✓ Nagłówek (nazwy) procedury, który umieszczony jest zaraz po spacji za słowem **procedure** (np. „**procedure** Sumuj”);
  - ✓ Parametry, które umieszczone są zaraz za nazwą procedury, pomiędzy nawiasami (np. „**procedure** Sumuj(A, B :Integer)”);
- Część opisową – deklaruje się tam stałe lokalne (po słowie **const**) lub zmienne lokalne (po słowie **var**);
- Część operacyjną, w której znajdują się instrukcje wykonujące określoną czynność – instrukcje te są umieszczone pomiędzy słowem „**begin**”, a słowem „**end**;”.

Poniżej przedstawiono przykład kilku definicji procedur.

```
uses Crt;

procedure JedenParametr(A :Integer); { Definicja procedury }
begin
  Writeln('Wprowadzono liczbę ', A, ', wynik (*): ', A*A);
{ Cześć operacyjna: Wyświetla tekst oraz wynik mnożenia dwóch liczb przechowywanych w
zmiennej liczbowej „A” }
end;

procedure DwaParametry(A, B :Integer); { Definicja procedury }
begin
  Writeln('Wprowadzono liczby ', A, ' i ', B, ', wynik (*): ', A*B);
{ Cześć operacyjna: Wyświetla tekst oraz wynik mnożenia dwóch liczb przechowywanych w
zmiennych „A” i „B” }
end;

procedure DwaParametryRef(A :Integer; var B :Integer);
{
 Parametr „B” procedury  „DwaParametryRef” poprzedzony jest słowem VAR, co umożliwia
dokonanie podmiany zawartości zmiennej „B”. Na przykład: A = 5, B = 10 i po wykonaniu
sumowania B = 15.
}
begin
  B:= A+B;
{ Dodawanie dwóch liczb przechowywanych w zmiennych „A” i „B” oraz przypisanie
zmiennej „B” wyniku z dodawania }
end;



var
  Licz :Integer; { Deklaracja zmiennej liczbowej }
begin
  ClrScr; { Czyści zawartość ekranu }
  JedenParametr(4); { Wywołanie procedury z jednym parametrem, tj. liczbą 4 }
```

```
  DwaParametry(5, 4); { Wywołanie procedury z dwoma parametrami, tj. liczbami 5 i 4 }

  Licz:= 0; { Wyzerowanie zmiennej liczbowej }
  Licz:= 10; { Przypisanie zmiennej liczbowej, liczby 10 }
  DwaParametryRef(5, Licz);

  Writeln('Wprowadzono liczby 5 i 10, wynik (+): ', Licz);
{
 Wyświetlenie tekstu, wraz z wynikiem, który jest przechowywany w zmiennej „Licz".
}
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.12. Stałe

Co to są stałe. Podaj przykłady ich zastosowania.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW06.PAS**.

Stałe wprowadza się do programu w celu ułatwienia modyfikacji parametru w programie. Możemy to wykonać w jednym miejscu. W przeciwnym przypadku należałoby wprowadzić zmiany w każdym miejscu, gdzie występuje ta stała. Stałe deklarujemy po słowie **const** (ang. Constants – stałe), a przed słowem **begin**. Rozróżnia się stałe globalne (widoczne w całym programie) oraz stałe lokalne (widoczne tylko w obrębie danej funkcji lub procedury, w której zostały zadeklarowane). Stała może przechowywać wartość liczbową, tekst lub wartość logiczną.

Poniższy program ilustruje zastosowanie stałych.

```
uses Crt;

const
{ Deklaracja stałych globalnych }
  Miesiace = 12; { Stała liczbowa przechowująca liczbę 12 }
  Doba = 24;

{ Stała tablicowa, przechowująca dni tygodnia }
  txtDniTygodnia :array[0..6] of String = ('Poniedziałek',
                        'Wtorek', 'Środa', 'Czwartek',
                        'Piątek', 'Sobota', 'Niedziela');

function Funkcja :String;
const
{ Deklaracja lokalnej stałej „txtTekst", przechowującej tekst }
  txtTekst = 'Wszystko jest możliwe pod warunkiem, że nie wiesz, o czym mówisz.';
begin
  Funkcja:= txtTekst; { Funkcja zwróci zawartość stałej „txtTekst" }
```

```
end;



begin
  ClrScr; { Czyści zawartość ekranu }
  Writeln('== Stałe ==');

{ Wyświetla tekst oraz zawartość stałej tablicowej o numerze 4 }
  Writeln('Dzien tygodnia: '+txtDniTygodnia[4]);

{ Wyświetla tekst oraz zawartość stałej „Doba" }
  Writeln('Doba ma ', Doba, ' godziny');

  Writeln(Funkcja); { Wywołanie funkcji „Funkcja" i wyświetlenie tekstu na ekranie }
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.13. Stałe w programie

Napisz program, który umożliwi przesunięcie wyświetlanego tekstu o określoną ilość znaków (np. o 7 kolumn w prawo i 7 wierszy w dół) – wykorzystując stałe.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW06A.PAS**.

```
uses Crt;

const
{ Deklaracja stałych }
  numLewo = 7;
  numGora = 7;
begin
  ClrScr; { Czyszczenie zawartości ekranu }
  Writeln; { Przesunięcie kursora o wiersz w dół }

{ Wyświetla tekst znajdujący się pomiędzy apostrofami, tj. znakami „ ' " na ekranie }
Writeln('== Stała ==');
Writeln;


  GotoXY(numLewo, numGora);
{
GotoXY(X, Y) – umieszcza tekst lub kursor w miejscu o współrzędnych X, Y. W tym
przykładzie tekst jest umieszczony we współrzędnych „numLewo+2" i „numGora"
}
```

```
{ Wyświetlenie tekstu o w miejscu o współrzędnych „numLewo” i „numGora” }
  Write('Turbo Pascal v7.0');
  GotoXY(numLewo+2, numGora+2);
  Write('Copyright(c)');
  GotoXY(numLewo+7, numGora+3);
  Write('by');
  GotoXY(numLewo+2, numGora+4);
  Write('Borland Team');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.14. Zmienne

Co to są zmienne. Podaj przykłady ich zastosowania.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW07.PAS**.

Zmienna umożliwia przechowywanie wartości liczbowej, tekstu lub wartości logicznej. Zmienne deklaruje się po słowie **var** (ang. Variables – zmienne), a przed słowem **begin**. Rozróżniamy zmienne globalne (widoczne w całym programie) i zmienne lokalne (widoczne w obrębie funkcji lub procedury, w której zostały zadeklarowane).

Poniższy program ilustruje zastosowanie zmiennych.

```
uses Crt;

var
{ Deklaracja zmiennych globalnych }
      Miesiac :Shortint; { Zmienna liczbowa }
  txtDniTygodnia :array[0..6] of String; { Zmienna tablicowa - tekstowa }


function Funkcja :String;
var
{ Deklaracja lokalnej zmiennej tekstowej „txtTekst” }
  txtTekst :String;
begin
  txtTekst:= 'Wszystko jest możliwe pod warunkiem, że nie wiesz, o czym mówisz.';
{ Przypisanie zmiennej „txtTekst” tekstu}

  Funkcja:= txtTekst; { Funkcja zwraca zawartość zmiennej „txtTekst” }
end;



begin
  ClrScr; { Wyczyszczenie zawartości ekranu }
```

```
  Writeln('== Stale ==');

  Writeln(Funkcja); { Wywołanie funkcji „Funkcja” i wyświetlenie tekstu na ekranie }

  Miesiac:= 9; { Przypisanie zmiennej liczby 9 }

{ Wyświetlenie tekstu oraz zawartości zmiennej „Miesiąc” }
  Writeln('Numer miesiaca: ', Miesiac);

  txtDniTygodnia[0]:= 'Poniedziałek'; { Przypisanie kolejnych elementów do tablicy }
  txtDniTygodnia[1]:= 'Wtorek';
  txtDniTygodnia[2]:= 'Środa';
  txtDniTygodnia[3]:= 'Czwartek';
  txtDniTygodnia[4]:= 'Piątek';
  txtDniTygodnia[5]:= 'Sobota';
  txtDniTygodnia[6]:= 'Niedziela';

{ Wyświetlenie tekstu oraz zawartości tablicy o numerze 4 }
  Writeln('Dzien tygodnia: '+txtDniTygodnia[4]);
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.15. Instrukcja warunkowa IF…THEN

Napisz program realizujący zasadę działania za pomocą instrukcji warunkowej IF…THEN.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW08.PAS** i **CW08A.PAS**.

Instrukcje warunkowe **IF...THEN** / **IF...THEN...ELSE** pozwalają na sterowanie programem, dzięki sprawdzeniu wartości wyrażenia logiczne. Jeżeli wyrażenie jest prawdziwe, to następuje wykonanie funkcji (procedur lub instrukcji) znajdujących się po słowie **THEN**.

UWAGA:
Jeżeli chce się wywołać większą ilość funkcji, procedur lub wpisać większą ilość instrukcji, to trzeba umieścić je pomiędzy słowami kluczowymi **begin** i **end**. W przypadku wywołania jednej funkcji, procedury lub wpisania jednej instrukcji, to słowa **begin** i **end** można pominąć.

Poniższy program ilustruje zastosowanie instrukcji warunkowych.

```
uses Crt;

var
  txtStr :String; { Deklaracja zmiennej tekstowej „txtStr” }
begin
  ClrScr; { Czyści zawartość ekranu }
```

```
  Write('Wpisz znak "a" lub "b":'+CHR(32));

  Readln(txtStr);
{ Odczytanie danych z klawiatury i przypisanie ich do zmiennej „txtStr” }

  if (txtStr = 'a') then Writeln('Wprowadzono znak: ', txtStr);
{
 Funkcja warunkowa IF…THEN umożliwia sprawdzenie warunku podanego pomiędzy
słowem IF, a słowem THEN. Jeżeli warunek jest spełniony, to zostaną wykonane instrukcje po
słowie THEN.

W tym przykładzie jest sprawdzany warunek równości litery „a” z zawartością zmiennej
„txtStr”. Jeżeli warunek będzie spełniony (tj. litera „a” będzie równa zawartości zmiennej
„txtStr”), to nastąpi wykonanie instrukcji znajdujących się po słowie THEN.

Po słowie THEN nie ma bloku BEGIN i END, ponieważ jest tu tylko jedna instrukcja. W
innym przypadku (tj. większej ilości instrukcji) należy instrukcje zamknąć w bloku BEGIN i
END.
}

  if (txtStr = 'b') then
  begin
    Writeln('Wprowadzono znak: ', txtStr);
    Writeln('To jest konstrukcja IF...THEN');
  end;
{
W tym przykładzie jest sprawdzany warunek równości litery „b” z zawartością zmiennej
„txtStr”. Jeżeli warunek będzie spełniony (tj. litera „b” będzie równa zawartości zmiennej
„txtStr”), to nastąpi wykonanie instrukcji znajdującej się po słowie THEN.

Instrukcje po słowie THEN znajdują się tu w bloku BEGIN i END, ponieważ tych instrukcji
jest więcej niż jedna.
}

end.
```

Poniższy program ilustruje zastosowanie instrukcji warunkowych.

```
uses Crt;

var
  A, B :Integer;
begin
  ClrScr; { Wyczyszczenie zawartości ekranu }
  Writeln; { Przesunięcie kursora o wiersz w dół }
  Writeln('Funkcja warunkowa IF..THEN - Dzielenie');
  Writeln;
  Write('Podaj liczbe 1:');
```

```
  Readln(A);
  Write('Podaj liczbe 2:');
  Readln(B);
  if (B <> 0) then
  begin
    Writeln('Podane liczby to ', A, ' i ', B);
    Writeln('Wynik dzielenia = ', A/B:1:2);
{
Zapis ten „A/B:1:2” umożliwia sformatowanie liczby w podanej postaci, np. 12.34 (tj. liczbę z
dwoma miejscami po przecinku). Parametry spełniają następującą role: „A/B” – przechowuje
wynik z dzielenia dwóch liczb przechowywanych w zmiennych „A” i „B”; „1” – określa
długość wyniku (tj. z ilu cyfr ma się składać łącznie z cyframi po przecinku - dzięki temu
wyniki znajdujące się w wierszach są zawsze wyrównywane do prawej); „2” – określa ilość
miejsc po przecinku.
}
  end
  else
    Writeln('Dzielenie jest niewykonalne!');
{
  if (warunek) then
  begin
    Instrukcja 1;
    Instrukcja 2;
    ……….;
    Instrukcja N;
  end
  else
  begin
    Instrukcja 1;
    Instrukcja 2;
    ……….;
    Instrukcja N;
  end;

Złożona instrukcja warunkowa IF…THEN…ELSE ma dwa bloki instrukcji.
Jeżeli zostanie spełniony warunek, to nastąpi wykonanie instrukcji po słowie THEN
(znajdujących się w bloku BEGIN i END). W przeciwnym przypadku nastąpi wykonanie
instrukcji po słowie ELSE.
}

end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.16. Wygaszacz ekranu

Napisz program, który będzie wyświetlał w losowo wybranych punktach ekranu dowolny napis.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW08B.PAS**.

```
uses Crt;

procedure Napis(X, Y, Color :Integer; txtNapis :String);
{Procedura wyświetla tekst o współrzędnych X, Y i w kolorze podanym w parametrze
COLOR}
begin
  { Napis }
  if (X < 1) then X:= 1;
{
 Jeżeli zawartość parametru X jest mniejsza od liczby 1, to przypisz temu parametrowi
wartość 1.
}

  if (Y < 1) then Y:= 1;
  GotoXY(X, Y); { Ustawia tekst na pozycji X, Y }
  TextBackground(BLACK); { Nadaje tłu kolor czarny }
  TextColor(Color); { Nadaje tekstowi kolor, który jest podany w parametrze COLOR }
  Writeln(txtNapis); { Wyświetla napis }

  GotoXY(1, 1); { Ustawia kursor na pozycji 1, 1 – Kursor }
  TextBackground(BLACK); { Nadaje tłu kolor czarny }
  TextColor(BLACK); { Nadaje tekstowi kolor czarny }

  Delay(222); { Wstrzymuje pracę programu na 222 milisekundy }
end;



const
  txtNapis = 'Turbo Pascal 7.0'; { Deklaracja stałej „txtNapis” przechowującej tekst }
var
  numX, numY :Shortint; { Deklaracja zmiennych liczbowych – całkowitych }
begin
  ClrScr; { Czyści zawartość ekranu }
  repeat
    Randomize; { Inicjuje generator liczb losowych }

    numX:= 0; { Wyzerowanie zmiennej }
    numX:= Random(64); { Losuje liczbę z zakresu od 0 do 64. Uwaga: Ekran składa się z 80
kolumn, minus 16 znaków na napis przechowywany w zmiennej „txtNapis”, co daje nam
liczbę 64. }

    numY:= 0;
    numY:= Random(24);

    Napis(numX, numY, WHITE, txtNapis);
{
```

```
 Wywołanie procedury NAPIS, która wyświetla tekst o współrzędnych „numX" i „numY",
oraz nadaje wyświetlanemu tekstowi kolor biały.
}

   Napis(numX, numY, LightGray, txtNapis);
   Napis(numX, numY, DarkGray, txtNapis);
   Napis(numX, numY, BLACK, txtNapis);
  until KeyPressed; { Dzięki pętli REPEAT…UNTIL tekst będzie wyświetlany na ekranie,
dopóki nie zostanie naciśnięty dowolny klawisz na klawiaturze. W momencie naciśnięcia
klawisza, wyświetlanie tekstu na ekranie zostanie przerwane. }
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.17. Instrukcja wyboru CASE

Napisz program, który prezentuje zasadę działania instrukcji wyboru CASE.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW08C.PAS**.

Instrukcja wyboru **CASE**, daje możliwość wybierania różnych wartości (lub wyświetlania różnych komunikatów), w zależności od zaistniałej sytuacji.

Na przykład:
Jeżeli zawartość wyrażenia „numCase" jest równa 4, to zostanie wyświetlony komunikat „Cyfra: 4". W momencie wpisania złej liczby, nastąpi wykonanie instrukcji wprowadzonej po słowie ELSE (tj. „Zła cyfra!").

```
uses Crt;

var
    numCase :Integer; { Deklaracja zmiennej liczbowej – całkowitej }
begin
  TextColor(WHITE); { Ustaw litery na kolor biały }
  ClrScr; { Czyści zawartość ekranu }

  Write('Podaj liczbę z zakresu 0..6:'+CHR(32)); { Wyświetla tekst oraz spację. }

  Readln(numCase); { Pobiera dane z klawiatury i przypisuje je do zmiennej liczbowej
„numCase" }

  case numCase of
   0: Writeln('Cyfra: 0');
   1: Writeln('Cyfra: 1');
   2: Writeln('Cyfra: 2');
   3: Writeln('Cyfra: 3');
   4: Writeln('Cyfra: 4');
```

```
    5, 6: Writeln('Cyfra: 5 lub 6');
  else
    Writeln('Zła cyfra!');
  end;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.18. Prosty kalkulator

Napisz prosty kalkulator wykonujący cztery podstawowe działania – przy pisaniu tego programu wykorzystaj instrukcję wyboru CASE.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW08D.PAS**.

```
uses Crt;

function jtbKalkulatorek(numWynik, numNowaLiczba :Real; chFunc :Char) :Real;
{ Funkcja wykonuje cztery podstawowe działania matematyczne (tj. dodawanie, odejmowanie,
mnożenie i dzielenie), a wynik tych działań przypisuje nazwie funkcji (tj. jtbKalkulatorek). }
begin
  { jtbKalkulatorek }
  case chFunc of
    '+': jtbKalkulatorek:= numWynik+numNowaLiczba; { Dodawanie }
    '-': jtbKalkulatorek:= numWynik-numNowaLiczba; { Odejmowanie }
    '*': jtbKalkulatorek:= numWynik*numNowaLiczba; { Mnożenie }
    '/': jtbKalkulatorek:= numWynik/numNowaLiczba; { Dzielenie }
  else
    jtbKalkulatorek:= numNowaLiczba; { Zawartość zmiennej „numNowaLiczba” jest
przypisana do nazwy funkcji, gdy żadne działanie nie będzie wybrane. }
  end;
{
Instrukcja wyboru CASE, daje możliwość wybierania różnych działań (w tym przykładzie
działań matematycznych), w zależności od podanego symbolu działania matematycznego.

Działanie funkcji polega na wybraniu odpowiedniej stałej (zakończonej znakiem „ : ”
dwukropka), na podstawie wartości przechowywanej w zmiennej „chFunc”.

Jeżeli zawartość zmiennej „chFunc” jest równa znakowi „+”, to zostanie wykonane działanie
dodawania dwóch zmiennych , przechowujących konkretne liczby. W momencie wpisania
złego znaku, nastąpi wykonanie instrukcji po słowie ELSE (tj. przypisanie zawartości
zmiennej „numNowaLiczba” do nazwy funkcji).
}

end;
```

```
begin
  ClrScr; { Wyczyszczenie zawartości ekranu }
  Writeln; { Przejście o jeden wiersz w dół }

  Writeln('== Prosty kalkulatorek ==');
  Writeln;

  Writeln; { Przejście o jeden wiersz w dół }

{ Wyświetlenie komunikatu, z jednoczesnym wywołaniem funkcji „jtbKalkulatorek(2, 3, '+')”.
Funkcja „jtbKalkulatorek(2, 3, '+')” dodaje dwie liczby (tj. 2, 3). }
  Writeln('Dodawanie liczb 2+3 : ', jtbKalkulatorek(2, 3, '+'):1:2);
{
Zapis ten „jtbKalkulatorek(2, 3, '+'):1:2” umożliwia sformatowanie liczby na postać, np.
12.34 (tj. liczbę z dwoma miejscami po przecinku). Parametry spełniają następującą funkcję:
„jtbKalkulatorek(2, 3, '+')” - przechowuje wynik z dodawania dwóch liczb; „1” – określa
długość wyniku (tj. z ilu cyfr ma się składać łącznie z cyframi po przecinku - dzięki temu
wyniki znajdujące się w wierszach są zawsze wyrównywane do prawej); „2” – określa ilość
miejsc po przecinku.
}

  Writeln;
  Writeln('Odejmowanie liczb 7-4 : ', jtbKalkulatorek(7, 4, '-'):1:2);

  Writeln;
  Writeln('Mnozenie liczb 5*5 : ', jtbKalkulatorek(5, 5, '*'):1:2);

  Writeln;
  Writeln('Dzielenie liczb 7/2 : ', jtbKalkulatorek(7, 2, '/'):1:2);
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.19. Pętla FOR…TO…DO / FOR…DOWNTO…DO

Napisz program, który ilustruje zasadę działania pętli FOR…TO…DO.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW09.PAS**.

Pętla **FOR…TO…DO / FOR…DOWNTO…DO** umożliwia wykonanie instrukcji (lub bloku instrukcji) z góry określoną ilość razy.

Do zatrzymania wykonywanej pętli służy instrukcja **BREAK**, która powoduje wcześniejsze zakończenie wykonywanej pętli.

Różnica między pętlą **FOR…TO…DO**, a pętlą **FOR…DOWNTO…DO** jest taka, że ta pierwsza (tj. **FOR…TO…DO**) powoduje odliczanie od wartości mniejszej do wartości większej (np. **for** TT:= 0 **to** 10 **do**). Natomiast druga pętla (tj. **FOR…DOWNTO…DO**)

powoduje odliczanie od wartości większej do wartości mniejszej (np. **for** TT:= 10 **downto** 0 **do**).

```
uses Crt;

var
  TT :Integer; { Deklaracja zmiennej liczbowej, typu całkowitego }
begin
  ClrScr; { Wyczyszczenie zawartości ekranu }
  Writeln; { Przesunięcie kursora o wiersz w dół }

  Writeln('Pętla FOR...TO/DOWNTO..DO');
  Writeln;


  { for...to...do }
  for TT:= 0 to 9 do
    Write(TT, ' '); { Wyświetlenie liczb od 0 do 9 }
  Writeln;

  { for...downto...do }
  for TT:= 9 downto 0 do
    Write(TT, ' '); { Wyświetlanie liczb od 9 do 0 }
  Writeln;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.20. Tablica ASCII

Napisz program wyświetlający znaki z tablicę kodów ASCII.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW09A.PAS**.

Tablica kodów ASCII (American Standard Code for Information Interchange) stanowi zestaw kodów używanych do reprezentacji znaków (liter, cyfr, znaków specjalnych [np. @, $, # itp.]). Każdy znak w tabeli ma przyporządkowaną wartość liczbową, np. litera „A” ma numer 65.
Tablica ASCII składa się 255 znaków, które podzielone są na kilka grup:

- Od 0 do 31 – znaki sterujące;
- Od 32 do 126 – znaki podstawowe;
- Od 127 do 255 – znaki dodatkowe (zawierają znaki graficzne, oraz znaki polskie itp.).

```
uses Crt;

var
  TT :Integer; { Deklaracja zmiennej liczbowej, typu całkowitego }
begin
```

```
  ClrScr; { Wyczyszczenie zawartości ekranu }

  Writeln('== Tablica ASCII ==');

  Writeln; { Przejście o jeden wiersz w dół }
  for TT:= 0 to 15 do
  begin
{ Pętla jest wykonywana 16 razy dla kolejnych TT (poczynając od 0 do 15) }

    Writeln('  ', TT+32, ' ', CHR(TT+32),
          '  ', TT+48, ' ', CHR(TT+48),
          '  ', TT+64, ' ', CHR(TT+64),
          '  ', TT+80, ' ', CHR(TT+80),
          '  ', TT+96, ' ', CHR(TT+96),
          '  ', TT+112, ' ', CHR(TT+112));
{ Wyświetlenie tablicy kodów ASCII.

Funkcja CHR(X) zwraca znak, którego kod ASCII określony jest w parametrze „X", np.
CHR(65) wyświetli znak „A". }

  end;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.21. Dodanie znaków pustych na początek tekstu

Napisz program umożliwiający dodanie spacji na początek tekstu.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW09B.PAS**.

```
uses Crt;

function fSpc(numSpc :Shortint) :String;
{ Funkcja zwraca określoną ilość spacji. Ilość tych znaków zależy od liczby przechowywanej
w parametrze „numSpc" }
var
    TT :Shortint; { Deklaracja zmiennej liczbowej, typu całkowitego }
  txtS :String; { Deklaracja zmiennej tekstowej }
begin
  { fSpc }
  txtS:= ''; { Wyczyszczenie zmiennej tekstowej }

  for TT:= 0 to numSpc-1 do
    txtS:= txtS+CHR(32); { Dodanie do zmiennej tekstowej znaków pustych w ilości określonej
przez parametr „numSpc". CHR(Numer_Znaku) – instrukcja umożliwiająca wyświetlenie
znaku o numerze podanym w parametrze „Numer_Znaku". W tym przykładzie jest to znak
spacji. }
```

```
  fSpc:= txtS; { Funkcja zwróci określoną ilość znaków pustych }
end;



const
  txtTekst = 'Turbo Pascal 7.0'; { Deklaracja stałej tekstowej i przypisanie do niej tekstu }
begin
  ClrScr; { Wyczyszczenie ekranu }

  Writeln('Program przesunie tekst w prawo o zadaną ilość pustych znaków.');

  Writeln; { Przejście o jeden wiersz w dół }

{ Wyświetlenie tekstu oraz zawartości stałej „txtTekst” }
  Writeln('Tekst źródłowy: '+txtTekst);

{ Wyświetlenie tekstu oraz zawartości stałej „txtTekst”, z dodanymi na początku znakami
pustymi. Ilość znaków pustych wynosi 12 }
  Writeln('Przesunięty tekst o zadaną liczbę spacji: '+fSpc(12)+txtTekst);
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

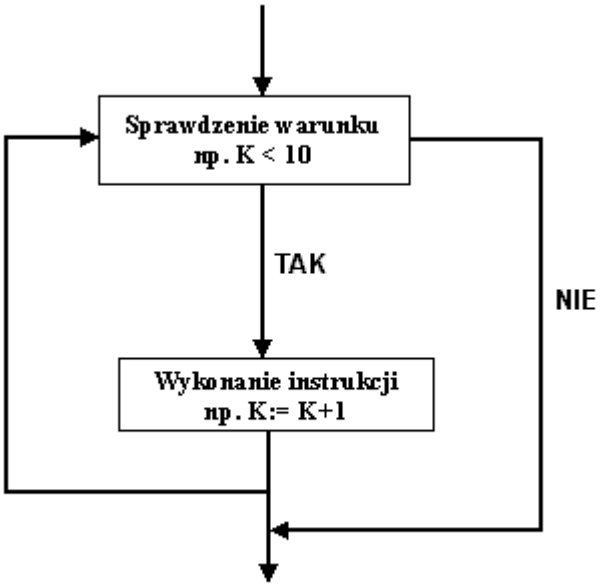
Ćwiczenie 3.22. Pętle WHILE...DO / REPEAT...UNTIL

Napisz program ilustrujący działanie pętli WHILE...DO i REPEAT...UNTIL.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW09C.PAS**.

Pętla **WHILE…DO** wykonuje blok instrukcji dopóty, dopóki spełniony jest podany warunek. Warunek ten jest sprawdzany na początku pętli przy każdym cyklu wykonania. W przypadku nie spełnienia warunku, wykonywanie pętli zostaje zatrzymane. Zdarzyć się może, że pętla nie zostanie wykonana ani razu, ponieważ warunek, który sprawdzany jest na początku przed wykonaniem pętli może mieć wartość FALSE (Fałsz).

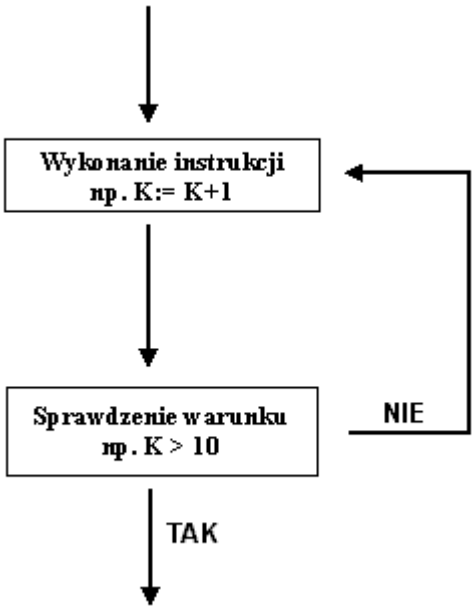
Schemat blokowy dla pętli **WHILE…DO**:



Jeżeli zmienna „K" jest mniejsza od liczby 10, to będzie warunek spełniony i nastąpi zwiększenie zmiennej „K" o wartość 1. Zmienna „K" będzie zwiększana, dopóki będzie spełniony warunek „K<10". W momencie gdy warunek przyjmie wartość FALSE (Fałsz), wykonanie pętli zostanie zatrzymane.

Natomiast pętla **REPEAT…UNTIL** wykonuje blok instrukcji do momentu spełnienia warunku. Warunek ten jest sprawdzany na końcu każdego cyklu. Sprawdzenie warunku na końcu cyklu sprawia, że pętla jest przynajmniej jeden raz wykonana.

Schemat blokowy dla pętli **REPEAT…UNTIL**:



Zwiększaj zmienną „K" o wartość 1 do momentu, gdy będzie ona większa od liczby 10. Gdy warunek zostanie spełniony, to przerwij wykonywanie pętli.

```
uses Crt;

var
  TT, K :Integer; { Deklaracja zmiennych liczbowych całkowitych }
begin
  ClrScr;


  { while...do }
  K:= 0; { Wyzerowanie zmiennej }
  while (K < 10) do
  begin
{ Pętla będzie wykonywana dopóki zmienna „K" będzie mniejsza od liczby 10 }
    K:= K+1; { Zwiększenie zmiennej „K" o wartość 1 }
  end;
  Writeln('Suma: ', K);


  { repeat...until }
  K:= 0;
  repeat
    K:= K+1; { Zwiększenie zmiennej „K" o wartość 1 }
  until (K > 10);
  Writeln('Suma: ', K);
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.23. Operatory

Wymień podstawowe operatory i napisz program ilustrujący ich działanie.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW10.PAS**.

Poniższa tabelka przedstawia wybrane operatory:

Wygląd	Znaczenie
:=	Przypisanie danych
=	Porównanie wartości dwóch typów
<>	Sprawdza, czy obie zmienne się różnią
<	Sprawdza, czy pierwsza wartość jest mniejsza od drugiej wartości
>	Sprawdza, czy pierwsza wartość jest większa od drugiej wartości
>=	Sprawdza, czy pierwsza wartość jest większa lub równa drugiej wartości
<=	Sprawdza, czy pierwsza wartość jest mniejsza lub równa drugiej wartości
and	Logiczne „i"
Or	Logiczne „lub"

Not	Zaprzeczenie

Poniższy program ilustruje zasadę działania operatorów.

```
uses Crt;

var
  txtStr1, txtStr2 :String; { Deklaracja zmiennych tekstowych }
begin
  ClrScr;
  Write('Wpisz znak 1 (np. "1" lub "2"):');
  Readln(txtStr1);

  Write('Wpisz znak 2 (np. "1" lub "2"):');
  Readln(txtStr2);

  { Operator: "=" }
  if (txtStr1 = txtStr2) then
    Writeln('Zawartość zmiennej "txtStr1" jest równa '+
          'zawartości zmiennej "txtStr2".');

  { Operator: "<>" }
  if (txtStr1 <> txtStr2) then
    Writeln('Zawartość zmiennej "txtStr1" jest różna od '+
          'zawartości zmiennej "txtStr2".');

  { Operator: "<" }
  if (txtStr1 < txtStr2) then
    Writeln('Zawartość zmiennej "txtStr1" '+
          'jest mniejsza od zawartości zmiennej "txtStr2".');

  { Operator: ">" }
  if (txtStr1 > txtStr2) then
    Writeln('Zawartość zmiennej "txtStr1" jest większa od '+
          'zawartości zmiennej "txtStr2".');

  { Operator: ">=" }
  if (txtStr1 >= txtStr2) then
    Writeln('Zawartość zmiennej "txtStr1" jest większa '+
          'od/lub równa zawartości zmiennej "txtStr2".');

  { Operator: "<=" }
  if (txtStr1 <= txtStr2) then
    Writeln('Zawartość zmiennej "txtStr1" jest mniejsza '+
          'od/lub równa zawartości zmiennej "txtStr2".');

  { Operator: AND }
  if ((txtStr1 = '1') and (txtStr2 = '1')) then
    Writeln('Zawartość zmiennej "txtStr1" jest równa 1 i '+
          'zawartość zmiennej "txtStr2" też jest równa 1.');
```

```
  { Operator: OR }
  if ((txtStr1 = '1') or (txtStr2 = '2')) then
    Writeln('Zawartość "txtStr1" jest równa 1 lub '+
          'zawartość zmiennej "txtStr2" jest równa 2.');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.24. Kod naciśniętego klawisza

Napisz program umożliwiający wyświetlenie kodu naciśniętego klawisza.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW11.PAS**.

```
uses Crt;

function PressAnyKey :Char;
{ Funkcja zwraca znak naciśniętego klawisza }
var
  Ch :Char; { Deklaracja zmiennej znakowej }
begin
  { KeyBoard }
  repeat until KeyPressed;
    Ch:= ReadKey;
    if (Ch = #0) then Ch:= ReadKey;
  PressAnyKey:= Ch; { Funkcja zwraca znak naciśniętego klawisza }
end;
{
 Funkcja „KeyPressed" służy do sprawdzenia, czy nastąpiło naciśnięcie na klawiaturze
dowolnego klawisza. Jeżeli klawisz został naciśnięty, to funkcja „KeyPressed" przyjmuje
wartość TRUE. W przeciwnym przypadku wartością funkcji jest FALSE.

Jeżeli wartością funkcji „KeyPressed" jest TRUE, to ponowne ustawienie tej funkcji na
wartość FALSE następuje poprzez użycie funkcji „ReadKey".

ReadKey – odczytuje znak naciśniętego klawisza, po czym następuje przypisanie tego znaku
do zmiennej „Ch".
Jeżeli wartością funkcji „KeyPressed" była wartość TRUE, to wartością funkcji „ReadKey"
będzie znak naciśniętego klawisza.

Konstrukcja „if (Ch = #0) then Ch:= ReadKey" umożliwia ponowne odczytanie znaku, w
przypadku naciśnięcia klawisza funkcyjnego (tj. klawisze od F1...F12), dzięki czemu zostaje
usunięty błąd w postaci komunikatu o naciśnięciu klawisza o kodzie zerowym.
}


var
```

```
  chChar :Char; { Deklaracja zmiennej znakowej }
begin
  ClrScr; { Czyści zawartość ekranu }
  Writeln; { Przesuwa kursor do następnego wiersza }

  Writeln('Wyświetl kod naciśniętego klawisza');

  repeat
    chChar:= PressAnyKey; { Przypisanie zmiennej „chChar" odczytanego znaku naciśniętego
klawisza }
    Writeln;

{ Wyświetlenie tekstu oraz znaku i numeru naciśniętego na klawiaturze klawisza }
    Writeln('Naciśnięty klawisz to "'+chChar+'" o kodzie ', ORD(chChar));
{ ORD(X) – zwraca numer porządkowy podanego znaku w parametrze X, np. ORD('A') = 65
}

  until (chChar = CHR(27)); {  Pętla ta, będzie wykonywana tak długo, dopóki nie nastąpi
naciśnięcie klawisza ESC. }
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 3.25. Wywołanie programu z parametrem

Napisz program umożliwiający wyświetlenie nazwy autora programu, dopiero po wywołaniu programu z parametrem.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW12.PAS**.

Wywołanie programu z parametrem, daje możliwość uruchomienia go z określonymi ustawieniami lub umożliwia wczytanie pliku z jednoczesnym uruchomieniem programu. Parametry należy podawać zaraz po nazwie programu, oddzielając ich od nazwy programu i innych parametrów znakiem spacji.

Z parametrami spotykamy się na co dzień, gdy np. chcemy oglądnąć jakiś plik graficzny i w momencie dwukrotnego kliknięcia na nim, następuje uruchomienie przeglądarki graficznej oraz automatyczne wyczytanie tegoż pliku (który jest właśnie parametrem) do niej.
Innym przykładem może być plik dokumentu programu MS WORD.
W momencie dwukrotnego kliknięcia na pliku dokumentu, uruchamia się edytor tekstu WORD z jednocześnie wczytanym plikiem (i ten plik jest parametrem).
Dzięki takiemu rozwiązaniu możliwe jest tworzenie powiązań plików o konkretnym rozszerzeniu z danym programem, np. pliki o rozszerzeniu PAS są powiązane z programem Borland DELPHI lub pliki o rozszerzeniu DOC są powiązane z programem MS WORD.

```
uses Crt;

begin
```

```
  ClrScr; { Czyści zawartość ekranu }

  Writeln('Podstawowy kurs Turbo Pascal`a 7.0');

   {
    Warunek "(ParamStr(1) = '/a')" sprawdza, czy zostały podane znaki „/a” w trakcie
uruchamiania programu.
Jeżeli znaki „/a” były podane w trakcie wywoływania programu, to następuje wykonanie
instrukcji po słowie THEN. W innym przypadku instrukcja po słowie THEN nie zostanie
wykonana.
   }
  if (ParamStr(1) = '/a') then
  begin
    Writeln('Copyright(c)by Jan T. Biernat');
  end;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Po skompilowaniu pliku o nazwie **CW12.PAS** otrzymujemy plik **CW12.EXE**.
Wykorzystując polecenie menu **Tryb MS-DOS**, w linii poleceń musisz wpisać nazwę programu i znaki „/a”, w celu wyświetlenia informacji o autorze programu.
Przykład: CW12.EXE /a (słowo „/a” jest parametrem) i po naciśnięciu klawisza ENTER, zostanie wyświetlona informacja o autorze programu. Gdyby parametr nie został podany, to na ekranie zostałaby wyświetlona tylko nazwa programu.

Ćwiczenie 3.26. Tablica

Napisz program, który wyświetli wprowadzone do tablicy liczby.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW13.PAS**.

```
uses Crt;

var
  Tablica :array[0..1, 0..1] of Integer; { Deklaracja tablicy dla liczb całkowitych }
    A, B :Integer; { Deklaracja zmiennych liczbowych całkowitych }
begin
  Tablica[0, 0]:= 1; { Przypisanie do elementu tablicowego o indeksach 0 i 0 liczby 1 }
  Tablica[0, 1]:= 2;
  Tablica[1, 0]:= 3;
  Tablica[1, 1]:= 4;

  ClrScr; { Wyczyszczenie zawartości ekranu }
  Writeln; { Przesunięcie kursora o jeden wiersz w dół }
  for A:= 0 to 1 do
    for B:= 0 to 1 do
```

```
    Write(Tablica[A, B], ' '); { Wyświetlenie elementu tablicy o indeksach znajdujących się w zmiennych A i B }
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

4. Ćwiczenia trudniejsze

W rozdziale tym przedstawiono ćwiczenia o większym stopniu trudności, które, mają na celu zapoznanie Czytelników ze sposobami rozwiązywania wybranych problemów.

Ćwiczenie 4.1. Konwersja liczb z postaci numerycznej na wartość tekstową i odwrotnie

Napisz program umożliwiający konwersję liczby na ciąg znaków i odwrotnie.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW14.PAS**.

Poniższe funkcje ilustrują sposoby konwersji liczby na ciąg znaków i odwrotnie.

```
uses Crt;

function IntToStr(X :Longint) :String;
{ Funkcja przekształca liczbę całkowitą na postać tekstową liczby (ciąg znaków) }
var
  txtText :String; { Deklaracja zmiennej tekstowej }
begin
  { IntToStr }
  STR(X, txtText);
{
  STR(X, txtText);
Procedura przekształca wartość liczbową wyrażenia (parametr „X") na reprezentację łańcuchową (parametr „txtText").
}
  IntToStr:= txtText; { Funkcja zwraca wartość tekstową }
end;

function StrToInt(txtStr :String) :Longint;
{ Funkcja przekształca postać tekstową liczby (ciąg znaków) na wartość liczbową, typu całkowitego }
var
   numNumber :Longint; { Deklaracja zmiennej liczbowej, typu całkowitego }
  numErrCode :Integer; { Deklaracja zmiennej liczbowej, typu całkowitego }
begin
  { StrToInt }
  VAL(txtStr, numNumber, numErrCode);
```

```
{
  VAL(txtStr, numNumber, numErrCode);
Procedura dokonuje przekształcenia wartości łańcuchowej (zmienna „txtStr") na
reprezentację numeryczną, zapamiętując wynik pod zmienną „numNumber". Jeżeli konwersja
przebiegła pomyślnie, to zmiennej „numErrCode" przypisywana jest wartość 0.
}

  StrToInt:= numNumber; { Funkcja zwraca wartość numeryczną, typu całkowitego }
end;

function StrToReal(txtStr :String) :Real;
{ Funkcja przekształca postać tekstową liczby (ciąg znaków) na liczbę rzeczywistą }
var
   numNumber :Real; { Deklaracja zmiennej liczbowej rzeczywistej }
  numErrCode :Integer; { Deklaracja zmiennej liczbowej całkowitej }
begin
  VAL(txtStr, numNumber, numErrCode);
{
  VAL(txtStr, numNumber, numErrCode);
Procedura dokonuje przekształcenia wartości łańcuchowej (zmienna „txtStr") na
reprezentację numeryczną, zapamiętując wynik pod zmienną „numNumber". Jeżeli konwersja
przebiegła pomyślnie, to zmiennej „numErrCode" przypisywana jest wartość 0.
}

  StrToReal:= numNumber; { Funkcja zwraca liczbę rzeczywistą }
end;

function RealToStr(X :Real) :String;
{ Funkcja przekształca liczbę rzeczywistą na postać tekstową liczby (ciąg znaków) }
var
  txtText :String;
begin
  STR(X:1:2, txtText);
{
  STR(X:1:2, txtText);
Procedura zamienia wartość liczbową wyrażenia (parametr „X") na ciąg znaków (parametr
„txtText").

Zapis ten „LICZBA:SZER:DEC" umożliwia sformatowanie liczby na postać, np. 12.34 (tj.
liczbę z dwoma miejscami po przecinku). Parametry spełniają następującą funkcję: LICZBA -
przechowuje wartość numeryczną; SZER – określa długość wyniku (tj. z ilu cyfr ma się
składać, łącznie z cyframi po przecinku - dzięki temu wyniki znajdujące się w wierszach są
zawsze wyrównywane do prawej); DEC – określa ilość miejsc po przecinku.

W tym przykładzie zapis „X:1:2" powoduje sformatowanie liczby na postać, np. 12.34 (tj.
liczbę z dwoma miejscami po przecinku).
}

  RealToStr:= txtText; { Funkcja zwraca wartość tekstową }
end;
```

```
begin
  ClrScr; { Czyszczenie zawartości ekranu }
  Writeln('Konwersja liczby 123 na tekst: '+IntToStr(123));
  Writeln('Konwersja tekstu "123" na liczbę: ', StrToInt('123'));
  Writeln('Konwersja tekstu "123.23" na liczbę: ', StrToReal('123.23'));
  Writeln('Konwersja liczby 123.23 na tekst: '+RealToStr(123.23));
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.2. Potęga, pierwiastek i losowanie liczby

Napisz program umożliwiający obliczenie potęgi n-tego stopnia i wyciągnięcie pierwiastka n-tego stopnia z podanych liczb.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW15.PAS**.

```
uses Crt;

function PotegaN(X, St :Real) :Real;
{ Funkcja oblicza potęgę n-tego stopnia }
begin
  PotegaN:= Exp(St*Ln(X)); { Obliczenie potęgi }
{
Exp(Liczba) – zwraca wartość funkcji e^x. Liczba „e" jest podstawą logarytmu naturalnego.
Liczba e w przybliżeniu wynosi 2.718281.

Ln(X) – oblicza logarytm naturalny z liczby X, np. Ln(e) = 1 – ponieważ liczba e = 2.718281.
}
end;

function PierwiastekN(X, St :Real) :Real;
{ Funkcja oblicza pierwiastek n-tego stopnia }
begin
   PierwiastekN:= Exp((1/St)*Ln(X)); { Funkcja zwraca obliczony pierwiastek }
end;

function RealToStr(X :Real) :String;
{ Funkcja przekształca liczbę rzeczywistą na postać tekstową liczby (ciąg znaków) }
var
  txtText :String;
begin
  STR(X:1:2, txtText);
  RealToStr:= txtText; { Funkcja zwraca wartość tekstową }
end;
```

```
begin
  ClrScr;
  Writeln('Pierwiastek 3-go stopnia z liczby 8 = '+RealToStr(PierwiastekN(8, 3)));
  Writeln('2^3 = '+RealToStr(PotegaN(2, 3)));
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.3. Wskaźnik postępu wykonywanego zadania

Napisz program, który wyświetli w procentach wskaźnik postępu wykonywanego zadania, na przykładzie pracy pętli FOR…TO…DO.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW16.PAS**.

W celu obliczanie postępu w procentach należy wykorzystać wzór P = Xz*100/Xs,
Gdzie: P – obliczony procent, Xz – wartość zmieniająca się, Xs – wartość stała.

Przykład: Jak procent stanowi liczba 25 z liczby 250.
Obliczenie: P = 25*100/250 = 10%

```
uses Crt;

function IntToStr(X :Longint) :String;
{ Funkcja przekształca liczbę całkowitą na postać tekstową liczby (ciąg znaków) }
var
  txtText :String; { Deklaracja zmiennej tekstowej }
begin
  { IntToStr }
  STR(X, txtText);
  IntToStr:= txtText; { Funkcja zwraca wartość tekstową }
end;

function Progress(Change, NoChange :Longint) :ShortInt;
{ Funkcja oblicza postęp wykonanego zadania w procentach }
begin
  if (NoChange = 0) then NoChange:= 1;
  Progress:= Round(Change * 100 / NoChange); { Obliczenie w procentach wykonanego
zadania i przypisanie wyniku do nazwy funkcji }
end;

function ShowProgress(Change, NoChange :Longint) :String;
{ Funkcja przekształca wartość liczbową na postać tekstową liczby (ciąg znaków) i wyświetla
wynik postępu }
begin
```

```
  ShowProgress:= IntToStr(Progress(Change, NoChange))+'%';
end;



const
  numMax = 99999; { Deklaracja stałej liczbowej }
var
  TT :Longint; { Deklaracja zmiennej liczbowej, typu całkowitego }
Begin
  ClrScr; { Wyczyszczenie zawartości ekranu }
  for TT:= 0 to numMax do
  begin
    GotoXY(7, 2); { Określenie współrzędnych wyświetlanego komunikatu }

{ Wyświetlenie informacji o procentowym postępie wykonywanego zadania }
    Writeln('Zadanie wykonane w '+ShowProgress(TT, numMax), ' (', TT, ')');
  end;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.4. Pliki – wykrywanie, usuwanie, zmiana nazwy

Napisz program umożliwiający wykrycie pliku oraz usunięcie i zmianę jego nazwy.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW17.PAS**.

```
uses Crt, Dos;
{
 Po słowie USES następuje deklaracja modułów, których instrukcje są wykorzystywane w
aktualnie pisanym programie.
Moduł CRT jest odpowiedzialny za obsługę ekranu i klawiatury.
Moduł DOS jest odpowiedzialny za wykonywanie funkcji i obsługę systemu operacyjnego.
}

function FileExists(txtFileName :String) :Boolean;
{ Funkcja sprawdza, czy plik o podanej nazwie w parametrze „txtFileName" fizycznie istnieje
na dysku }
var
  Info :SearchRec; { Deklaracja zmiennej „Info" typu „SearchRec" }
{
Typ SearchRec jest rekordem posiadającym m.in. pole „Name". Typ SearchRec jest
stosowany w procedurach „FindFirst" i „FindNext".

Rekord jest to zbiór informacji dotyczących jednego tematu. Rekord wyświetlany jest w formie
wierszy, a każde pole danych tworzy kolumnę.
```

Poniższa tabela przedstawia dwa rekordy:

LP	*Nazwa pliku*	*Rozmiar*	*Czas utworzenia*
1	*czytaj.txt*	*12 456*	*2004-04-08 12:45*
2	*Nc.com*	*2 311*	*1989-01-23 10:00*

}
begin
{ FileExists }
FindFirst(txtFileName, Directory, Info);
{ Procedura FindFirst przeszukuje zbiory w podanym katalogu. Nazwa katalogu i pliku jest podana w parametrze „txtFileName". Po odnalezieniu szukanego zbioru, informacja o tym jest przekazana do zmiennej typu „SearchRec". Wynik szukania zbioru jest przekazywany do zmiennej „DosError". Jeżeli wyszukanie zbioru zakończyło się powodzeniem, to do zmiennej „DosError" zostaje przypisana wartość 0. }

FileExists:= FALSE; *{ Funkcja zwraca wartość FALSE }*
if (DosError = 0) **then** FileExists:= TRUE; *{ Jeżeli plik o podanej nazwie istnieje fizycznie na dysku, to funkcja zwróci wartość TRUE }*
end;

function DeleteFile(txtFileName :**String**) :Boolean;
{ Funkcja usuwa plik o podanej nazwie }
var
TF :Text; *{ Deklaracja zmiennej TF, określającej plik tekstowy }*
begin
{ DeleteFile }
DeleteFile:= FALSE; *{ Funkcja zwraca wartość FALSE }*

if (FileExists(txtFileName) = TRUE) **then**
begin
{ Jeżeli plik istnieje fizycznie na dysku, to wykonaj poniższe instrukcje }

Assign(TF, txtFileName); *{ Skojarzenie pliku o nazwie podanej w parametrze „txtFileName" ze zmienną plikową „TF" }*

SetFAttr(TF, Archive); *{ Nadanie plikowi skojarzonemu ze zmienną plikową „TF" atrybutu „Archive" (archiwalnego) }*

Erase(TF); *{ Usunięcie pliku skojarzonego ze zmienną plikową „TF" }*
DeleteFile:= TRUE; *{ Funkcja zwróci wartość TRUE }*
end;
end;

function RenameFile(txtFileNameOLD, txtFileNameNEW :**String**) :Boolean;
{ Funkcja zamienia starą nazwę pliku, na nową nazwę }
var
TF :Text; *{ Deklaracja zmiennej TF, określającej plik tekstowy }*
begin
{ RenameFile }
RenameFile:= FALSE; *{ Funkcja zwraca wartość FALSE }*
if (FileExists(txtFileNameNEW) = FALSE) **then**

```
  begin
{ Jeżeli nie ma pliku o nazwie podanej w parametrze „txtFileNameNEW" fizycznie na dysku,
to wykonaj poniższe instrukcje }
    if (FileExists(txtFileNameOLD) =TRUE) then
    begin
{ Jeżeli plik o nazwie podanej w parametrze „txtFileNameOLD" znajduje się fizycznie na
dysku, to wykonaj poniższe instrukcje }

      Assign(TF, txtFileNameOLD); { Skojarzenie pliku o nazwie podanej w parametrze
„txtFileNameOLD" ze zmienną plikową „TF" }

      Rename(TF, txtFileNameNEW); { Nadanie plikowi skojarzonemu ze zmienną plikową
„TF" nowej nazwy, podanej w parametrze „txtFileNameNEW" }

      SetFAttr(TF, Archive+ReadOnly); { Nadanie plikowi skojarzonemu ze zmienną plikową
„TF" atrybutu „Archive" (archiwalnego) i atrybutu „ReadOnly" (tylko do odczytu) }

      RenameFile:= TRUE; { Funkcja zwróci wartość TRUE }
    end;
  end;
end;



var
  okPlik :Boolean; { Deklaracja zmiennej logicznej }
begin
  ClrScr; { Wyczyszczenie zawartości ekranu }
  Writeln; { Przejście o jeden wiersz w dół }

  Writeln('== Pliki - Wykrywanie, Usuwanie, Zmiana nazwy ==');
  Writeln;

  Write('Zmiana nazwy pliku: ');
  okPlik:= RenameFile('plik.txt', 'pas.txt'); { Zamiana nazwy pliku z nazwy PLIK.TXT na
nazwę PAS.TXT }
  if (okPlik = TRUE) then
    Write('Nazwa pliku została zmieniona.') { Wyświetl informację o dokonanej zamianie }
  else
    Write('Brak pliku o podanej nazwie!');


  Writeln;
  Write('Usuwanie pliku: ');
  okPlik:= DeleteFile('pas.txt'); { Usunięcie pliku o nazwie PAS.TXT }
  if (okPlik = TRUE) then
    Write('Plik został usunięty.')
  else
    Write('Brak pliku o podanej nazwie!');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.5. Zapisanie danych do pliku tekstowego

Napisz program umożliwiający zapisanie danych do pliku tekstowego.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW18.PAS**.

```
uses Crt, Dos;

function FileExists(txtFileName :String) :Boolean;
{ Funkcja sprawdza, czy plik o podanej nazwie w parametrze „txtFileName" fizycznie istnieje na dysku }
var
  Info :SearchRec; { Deklaracja zmiennej „Info" typu „SearchRec" }
begin
  { FileExists }
  FindFirst(txtFileName, Directory, Info);

  FileExists:= FALSE; { Funkcja zwraca wartość FALSE }
  if (DosError = 0) then FileExists:= TRUE; { Jeżeli plik o podanej nazwie istnieje fizycznie na dysku, to funkcja zwróci wartość TRUE }
end;

function FILE_jbTextSaveToFile(txtFileName, txtText: String): Boolean;
{ Funkcja zapisuje dane zawarte w parametrze „txtText" do pliku tekstowego, którego nazwa jest przechowywana w parametrze „txtFileName" }
var
  TF :Text; { Deklaracja zmiennej TF, określającej plik tekstowy }
begin
  { TextSaveToFile }
  if (FileExists(txtFileName) = TRUE) then
  begin
{ Jeżeli plik o podanej nazwie istnieje, to dopisz do niego nowy wiersz z danymi }

    Assign(TF, txtFileName); { Skojarzenie pliku o nazwie podanej w parametrze „txtFileName" ze zmienną plikową „TF" }

    Append(TF); { Otwiera plik skojarzony ze zmienną „TF" i umożliwia dodanie do niego nowego wiersza. Procedury APPEND można używać tylko dla plików tekstowych }

    Writeln(TF, txtText); { Zapisanie nowej linii do pliku skojarzonego ze zmienną „TF" }
    Close(TF); { Zamknięcie pliku skojarzonego ze zmienną „TF" }
  end
  else
  begin
{ Jeżeli plik o podanej nazwie nie istnieje, to wykonaj poniższe instrukcje }
```

```
    Assign(TF, txtFileName);
    Rewrite(TF); { Tworzy nowy fizyczny plik o nazwie skojarzonej ze zmienną plikową „TF” }
    Writeln(TF, txtText); { Zapisanie nowego wiersza z danymi do pliku skojarzonego ze
zmienną „TF” }
    Close(TF);
  end;
  FILE_jbTextSaveToFile:= TRUE; { Funkcja zwraca wartość TRUE }
end;



const
  txtNazwaPliku = 'dane.txt'; { Deklaracja stałej tekstowej „txtNazwaPliku” przechowującej
nazwę pliku }
begin
  ClrScr; { Wyczyszczenie ekranu }
  Writeln; { Przejście do nowego wiersza }

  Writeln('== Zapisywanie danych do pliku tekstowego ==');
  Writeln;
  FILE_jbTextSaveToFile(txtNazwaPliku, 'Podstawowy kurs'); { Zapisanie danych do pliku
tekstowego }

  FILE_jbTextSaveToFile(txtNazwaPliku, 'Turbo Pascal`a 7.0');
  FILE_jbTextSaveToFile(txtNazwaPliku, 'Copyright(c)');
  FILE_jbTextSaveToFile(txtNazwaPliku, 'by');
  FILE_jbTextSaveToFile(txtNazwaPliku, 'Jan-Tadeusz Biernat');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.6. Odczytanie danych z pliku tekstowego

Napisz program umożliwiający odczytanie danych z pliku tekstowego i wyświetlenie ich na ekran.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW18A.PAS**.

```
uses Crt, Dos;

function FileExists(txtFileName :String) :Boolean;
{ Funkcja sprawdza, czy plik o podanej nazwie w parametrze „txtFileName” fizycznie istnieje
na dysku }
var
  Info :SearchRec; { Deklaracja zmiennej „Info” typu „SearchRec” }
begin
  { FileExists }
```

```
  FindFirst(txtFileName, Directory, Info);

  FileExists:= FALSE; { Funkcja zwraca wartość FALSE }
  if (DosError = 0) then FileExists:= TRUE; { Jeżeli plik o podanej nazwie istnieje fizycznie
na dysku, to funkcja zwróci wartość TRUE }
end;

procedure OdczytajDane;
{ Procedura umożliwia odczytanie zawartości pliku tekstowego oraz wyświetlenie
odczytanych danych na ekran }
const
  txtNazwaPliku = 'dane.txt'; { Deklaracja stałej „txtNazwaPliku" przechowującej nazwę
pliku }
var
  txtCzytaj :String; { Deklaracja zmiennej tekstowej }
        FT :Text; { Deklaracja zmiennej TF, określającej plik tekstowy }
begin
  Assign(FT, txtNazwaPliku); { Skojarzenie pliku o nazwie podanej w parametrze
„txtFileName" ze zmienną plikową „TF" }

  Reset(FT); { Otwarcie pliku tekstowego skojarzonego ze zmienną plikową „TF" }
  while not EOF(FT) do
  begin
{ Pętla wykonywana jest do momentu osiągnięcia końca pliku.
 O tym, że plik osiągnął swój koniec informuje funkcja EOF, która jest skojarzona ze zmienną
plikową „TF". }

    txtCzytaj:= ''; { Wyzerowanie zmiennej }
    ReadLn(FT, txtCzytaj); { Odczytanie kolejnego wiersza z pliku skojarzonego ze zmienną
plikową „TF" }
    Writeln(txtCzytaj); { Wyświetlenie na ekranie odczytanej z pliku informacji }
  end;
  Close(FT); { Zamknięcie pliku skojarzonego ze zmienną plikową „TF" }
end;



begin
  ClrScr;
  Writeln;
  Writeln('== Odczytanie danych z pliku tekstowego ==');
  Writeln;
  OdczytajDane; { Wywołanie procedury odczytującej dane z pliku tekstowego }
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.7. Rysowanie trójkąta

Napisz program, który narysuje trójkąt wykorzystując znaki „”.*

Sposób wykonania

Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW19.PAS**.

```
uses Crt;

function fAscii(numSpc :Shortint; chChar :Char) :String;
{ Funkcja zwraca łańcuch wypełniony znakami mnożenia, tj. „*” }
var
   TT :Shortint; { Deklaracja zmiennej liczbowej całkowitej }
  txtS :String; { Deklaracja zmiennej tekstowej }
begin
  { fAscii }
  txtS:= ''; { Wyczyszczenie zmiennej tekstowej }

  for TT:= 0 to numSpc-1 do
    txtS:= txtS+chChar; { Dodanie do zmiennej „txtS” dowolnego znaku tekstowego podanego
w parametrze „chChar”. Ilość dodanych znaków zależy od liczby przechowywanej w
parametrze „numSpc” }

  fAscii:= txtS; { Przypisanie nazwie funkcji określonej ilości znaków. Ilość znaków jest
przechowywana w parametrze „numSpc”. }
end;



var
  TT :Shortint; { Deklaracja zmiennej liczbowej całkowitej }
begin
  ClrScr; { Wyczyszczenie ekranu }
  Writeln('Program rysuje trójkąt.');

  Writeln; { Przejście o wiersz w dół }

{ Wyświetlenie trójkąta na ekranie }
  for TT:= 0 to 9 do
  begin
    Write(fAscii(9-TT, CHR(32))+fAscii(TT+1, '*')+fAscii(TT, '*'));
    Writeln;
  end;

  Writeln;

{ Wyświetlenie trójkąta na ekranie }
  for TT:= 9 downto 0 do
  begin
    Write(fAscii(9-TT, CHR(32))+fAscii(TT+1, '*')+fAscii(TT, '*'));
    Writeln;
```

```
  end;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.8. Inicjały

Napisz program, który wyciągnie z podanych wyrazów inicjały.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW20.PAS**.

```
uses Crt;

function Inicjaly(txtStr :String) :String;
{ Funkcja zwraca inicjały pobrane z tekstu podanego w parametrze „txtStr" }
var
      TT :Integer; { Deklaracja zmiennej liczbowej, typu całkowitego }
  txtTemp :String; { Deklaracja zmiennej tekstowej }
begin
   Inicjaly:= ''; { Funkcja zwraca wartość pustą }
   if (txtStr<>'') then
   begin
{ Jeżeli zmienna „txtStr" posiada jakąś wartość tekstową, to wykonaj poniższe instrukcje  }

     txtTemp:= ''; { Czyści zmienną tekstową }
     for TT:= 1 to Length(txtStr) do
       if (txtStr[TT-1] = CHR(32)) then txtTemp:= txtTemp+txtStr[TT];
{
 FOR TT:= 1 TO Length(txtStr) DO – jest wykonywana tyle razy, ile znaków znajduje się w
parametrze „txtStr".

Jeżeli będzie spełniony warunek „txtStr[TT-1] = CHR(32)", tj. znak na pozycji TT-1 będzie
znakiem spacji (tj. znakiem pustym) , to nastąpi pobranie następnej litery (tj. litery
znajdującej się na pozycji TT).

Length() - Oblicza z ilu znaków składa się wprowadzony tekst
 }

     Inicjaly:= txtStr[1]+txtTemp;
 { Pobiera znak pierwszy „txtStr[1]", dodając do niego wcześniej pobrane znaki znajdujące
się w zmiennej „txtTemp" oraz przypisuje wszystkie znaki nazwie funkcji. }
   end;
end;



const
  txtTekst = 'Borland Turbo Pascal'; { Deklaracja stałej tekstowej, przechowującej jakiś tekst }
```

```
begin
  ClrScr; { Czyści zawartość ekranu }
  Writeln('== Inicjaly ==');

  Writeln('Podany tekst: '+txtTekst); { Wyświetlenie oryginalnego tekstu }

  Writeln('Inicjaly: '+Inicjaly(txtTekst)); { Wyświetlenie na ekranie pobranych inicjałów }
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.9. Wyselekcjonowanie liczb z tekstu

Napisz program, który wyciągnie z podanego tekstu tylko cyfry i znak minus.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW21.PAS**.

```
uses Crt;

function SameCyfry(txtStr :String) :String;
{
 Funkcja wybiera z cyfry i znak minus z tekstu przechowywanego w parametrze „txtStr”.

 Przykład:
  1) wprowadzony tekst "-Przy123kła-d4owy te5k6,st 7.8"
  2) Wyselekcjonowany tekst "-123456,78"
}
var
   TT :Shortint; { Deklaracja zmiennej liczbowej, typu całkowitego }
  txtS :String; { Deklaracja zmiennej tekstowej }
begin
  { SameCyfry }
  txtS:= ''; { Wyczyszczenie zmiennej tekstowej }

  for TT:= 1 to Length(txtStr) do
    if (txtStr[TT] in ['0'..'9', '-']) then txtS:= txtS+txtStr[TT];
    {
     FOR TT:= 1 TO Length(txtStr) DO – jest wykonywana tyle razy, ile znaków znajduje się w
parametrze „txtStr”.

     Length() - Oblicza z ilu znaków składa się wprowadzony tekst.

     IF (txtStr[AA] in ['0'..'9', '-']) THEN
Sprawdza czy znak jest zgodny ze znakami od 0 do 9 oraz znakiem minus.
Konstrukcja ta zwalnia programistę ze żmudnego wpisywania warunku dla każdej cyfry (w
postaci: IF (txtStr[AA] = '0') THEN, IF (txtStr[AA] = '1') THEN,..., IF (txtStr[AA] = '9')
THEN).
```

```
    txtS:= txtS+txtStr[TT];
```
Dodanie do zmiennej "txtS" znaku od 0 do 9 i znaku minus (znak ten pobrany jest ze zmiennej „txtStr", z pozycji o numerze indeksu przechowywanym w zmiennej TT), w przypadku spełnienia warunku.
```
   }

 SameCyfry:= txtS; { Funkcja zwraca tylko liczby i znak minus }
end;



const
  txtTekst = '1abc 2de 3fghijklm5n,-6'; { Deklaracja stałej tekstowej przechowującej tekst }
begin
  ClrScr; { Czyści zawartość ekranu }
  Writeln('Program wyciąga z tekstu tylko cyfry.');
  Writeln;
  Writeln('Tekst źródłowy: '+txtTekst);
  Writeln('Wyciągnięte liczby: '+SameCyfry(txtTekst));
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.10. Zmiana liter na małe

Napisz program, który zamieni wszystkie litery na małe.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW22.PAS**.

```
uses Crt;

function LITERY_Male(txtStr :String) :String;
{ Funkcja zamienia wszystkie znaki tekstu przechowywanego w parametrze „txtStr" na małe znaki }

  function MalyZnak(chZnak :Char) :Char;
{ Funkcja zamienia duży znak na mały znak }
  var
    TT :Byte; { Deklaracja zmiennej liczbowej, typu całkowitego }
  begin
    for TT:= 65 to 90 do
      if (Chr(TT) = chZnak) then chZnak:= Chr(TT+32);
{
Jeżeli będzie spełniony warunek „Chr(TT) = chZnak", tj. podana duża litera przechowywana w parametrze „chZnak" będzie równa literze zwróconej przez funkcję „Chr", to przypisz zmiennej „chZnak" małą literę. Na przykład: Została wprowadzona duża litera „A", a funkcja zamieni ją na małą literę „a".
}
```

```
    MalyZnak:= chZnak; { Funkcja zwraca małą literę }
  end;

var
  I :Byte; { Deklaracja zmiennej liczbowej, typu całkowitego }
begin
  for I:= 1 to Length(txtStr) do
    txtStr[I]:= MalyZnak(txtStr[I]);
{
 Pętla „for I:= 1 to Length(txtStr) do" będzie wykonywana tyle razy, ile znaków znajduje się
w parametrze „txtStr".

„txtStr[I]" – Zapis ten umożliwia pobranie znaku z tekstu i po zamianie umieszcza go w tym
samym miejscu ciągu znaków. Zmienna „I" określa pozycję znaku w tekście.
}

  LITERY_Male:= txtStr; { Funkcja zwróci tekst pisany małymi literami }
end;



const
  txtTekst = 'Nie ma rzeczy niemożliwych dla kogoś, kto nie musi ich sam robić.';
begin
  ClrScr;
  Writeln('Tekst w normalnej postaci: '+txtTekst);
  Writeln('Tekst pisany małymi literami: '+LITERY_Male(txtTekst));
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.11. Wyrazy pisane z dużej litery

Napisz program, który umożliwi zamianę pierwszej litery każdego wyrazu na dużą.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW23.PAS**.

```
uses Crt;

function DuzyZnak(chZnak :Char) :Char;
{ Funkcja zamienia mały znak na duży }
var
  TT :Byte; { Deklaracja zmiennej liczbowej, typu całkowitego }
begin
  for TT:= 97 to 122 do
    if (Chr(TT) = chZnak) then chZnak:= Chr(TT-32);
{
```

Jeżeli będzie spełniony warunek „Chr(TT) = chZnak", tj. podana mała litera przechowywana w parametrze „chZnak" będzie równa literze zwróconej przez funkcję „Chr", to przypisz zmiennej „chZnak" dużą literę. Na przykład: Została wprowadzona mała litera „a", a funkcja zamieni ją na dużą literę „A".
}
```
  DuzyZnak:= chZnak; { Funkcja zwraca dużą literę }
end;

function WyrazDuzaLitera(txtStr :String; chrFind :Char) :String;
{ Funkcja zamienia pierwszą literę każdego wyrazu na dużą literę }
var
      TT :Integer; { Deklaracja zmiennej liczbowej, typu całkowitego }
  txtTemp :String; { Deklaracja zmiennej tekstowej }
begin
  WyrazDuzaLitera:= ''; { Funkcja zwróci znak pusty }
  if (txtStr<>'') then
  begin
{ Jeżeli zmienna „txtStr" zawiera jakąś wartość tekstową, to wykonaj poniższe instrukcje }

    txtTemp:= ''; { Wyzerowanie zmiennej tekstowej }
    for TT:= 2 to Length(txtStr) do
      if (txtStr[TT-1] = CHR(32)) or
         (txtStr[TT-1] = chrFind) then
        txtTemp:= txtTemp+DuzyZnak(txtStr[TT])
      else
        txtTemp:= txtTemp+txtStr[TT];
{
 FOR TT:= 2 TO Length(txtStr) DO jest wykonywana tyle razy, ile jest znaków znajduje się w wprowadzonym tekście.

Jeżeli będzie spełniony warunek „txtStr[TT-1] = CHR(32)" lub „txtStr[TT-1] = chrFind", tj. znak na pozycji TT-1 będzie znakiem spacji lub będzie znakiem podanym w parametrze „chrFind", to powiększ następną literę (tj. literę znajdującą się na pozycji „TT". ).

Length() - Oblicza z ilu znaków składa się wprowadzony tekst
 }

    WyrazDuzaLitera:= DuzyZnak(txtStr[1])+txtTemp;
{ Pobiera znak pierwszy „DuzyZnak(txtStr[1])", dodając do niego wcześniej pobrane znaki znajdujące się w zmiennej „txtTemp" oraz przypisuje wszystkie znaki nazwie funkcji. }
  end;
end;



const
  txtTekst = 'borland turbo pascal by borland-team';
begin
  ClrScr;
  Writeln('== Każdy wyraz rozpoczyna się z dużej litery ==');
```

```
  Writeln('Podany tekst: '+txtTekst);
  Writeln('Każdy wyraz rozpoczyna się z dużej litery: '+WyrazDuzaLitera(txtTekst, '-'));
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.12. Zamiana jednego ciągu znaków na inny

Napisz program, który umożliwi zamianę jednego ciągu znaków na inny ciąg znaków.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW24.PAS**.

```
uses Crt;

function jtbZnajdzZamien(txtText, txtFind, txtReplace :String) :String;
{ Funkcja zamienia ciąg znaków na inny ciąg znaków }
var
  numPos, numLen: Integer; { Deklaracja zmiennych liczbowych całkowitych }
begin
  { jtbZnajdzZamien }

  numLen:= 0; { Wyzerowanie zmiennej liczbowej }
  numLen:= Length(txtFind); { Obliczenie długości ciągu znaków }

  {
   Pos() – Zwraca pozycję znalezionego ciągu znaków.

   Pos(Szukany_Tekst, Tekst_w_Którym_się_Szuka)
   Funkcja zwraca pozycję występowania pierwszego znaku podłańcucha (parametr
   „Szukany_Tekst”) w ciągu znaków (parametr „Tekst_w_Którym_się_Szuka”).
  }
  while (Pos(txtFind, txtText) > 0) do
  begin
    {
     Pętla jest wykonywana tak długo, jak długo będzie występował
     wyszukiwany ciąg znaków. W przypadku nie znalezienia szukanego
     ciągu znaków, pętla nie wykona się ani razu.
    }

    numPos:= 0; { Wyzerowanie zmiennej liczbowej }

    numPos:= Pos(txtFind, txtText); { Przypisanie zmiennej „numPos” numeru pozycji
pierwszego znalezionego tekstu w ciągu znaków. }

    Delete(txtText, numPos, numLen); { Usunięcie znalezionego tekstu (ciągu znaków) }
    Insert(txtReplace, txtText, numPos); { Wstawienie nowego tekstu (ciągu znaków) w miejsce
starego }
  end;
```

```
  jtbZnajdzZamien:= txtText; { Funkcja zwróci tekst po dokonanych zmianach }
end;



const
  txtTekst = 'Autorem programu jest Jan Tadeusz Biernat';
begin
  ClrScr;
  Writeln('Tekst: "'+txtTekst+'"');
  Writeln;
  Writeln('Program zamieni wyraz "Tadeusz" na wyraz "T."');
  Writeln;
  Writeln('Tekst po zamianie: "'+jtbZnajdzZamien(txtTekst, 'Tadeusz', 'T.')+'"');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.13. Zapisanie danych do pliku rekordowego

Napisz program, który umożliwi zapisanie danych podanych z klawiatury do pliku rekordowego.

Sposób wykonania

Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW25.PAS**.

```
uses Crt, Dos;
{
 Po słowie USES następuje deklaracja modułów, których instrukcje są wykorzystywane w
aktualnie pisanym programie.
Moduł CRT jest odpowiedzialny za obsługę ekranu i klawiatury.
Moduł DOS jest odpowiedzialny za wykonywanie funkcji i obsługę systemu operacyjnego.
}

type
{ Struktura pliku rekordowego }
  TDane = record
    Nazwisko :String[60];
       Imie :String[30];
       Wiek :String[3];
  end;

var
    RF :file of TDane; { Deklaracja zmiennej rekordowej „RF” }
  Dane :TDane; { Deklaracja obiektu „Dane” }
begin
  ClrScr; { Czyści zawartość ekranu }
  Writeln; { Przejście o jeden wiersz w dół }
  Writeln('== Plik rekordowy - zapisywanie ==');
```

```
  Writeln;

  Assign(RF, 'dane.dat'); { Skojarzenie pliku o nazwie DANE.DAT ze zmienną plikową „RF” }
  Rewrite(RF); { Utworzenie nowego pliku o nazwie skojarzonej ze zmienną plikową „RF” }
  repeat
    Write('Podaj nazwisko:'); { Wyświetla pytanie }
    Readln(Dane.Nazwisko); { Pobiera dane z klawiatury i przypisuje ją do obiektu
„Dane.Nazwisko” }

    Write('Podaj imie:');
    Readln(Dane.Imie);

    Write('Podaj wiek:');
    Readln(Dane.Wiek);

    if (Dane.Nazwisko<>'') then Write(RF, Dane); { Zapisz dane do pliku, gdy obiekt
„Dane.Nazwisko” będzie zawierał tekst. W przeciwnym razie nie wykonuj zapisu. }

    Writeln;
  until (Dane.Nazwisko = ''); { Powoduje wyjście z pętli, gdy obiekt „Dane.Nazwisko” nie
posiada żadnego tekstu }
  Close(RF); { Zamknij plik skojarzony ze zmienną „RF” }
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.14. Odczytanie danych z pliku rekordowego

Napisz program, który umożliwi odczytanie danych z pliku rekordowego oraz wyświetlenie ich na ekran.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW25A.PAS**.

```
uses Crt, Dos;

type
{ Struktura pliku rekordowego }
  TDane = record
    Nazwisko :String[60];
       Imie :String[30];
       Wiek :String[3];
  end;

var
    RF :file of TDane; { Deklaracja zmiennej rekordowej „RF” }
  Dane :TDane; { Deklaracja obiektu „Dane” }
begin
  ClrScr; { Czyści zawartość ekranu }
```

```
  Writeln; { Przejście o jeden wiersz w dół }
  Writeln('== Plik rekordowy - odczytywanie ==');
  Writeln;

  Assign(RF, 'dane.dat'); { Skojarzenie pliku o nazwie DANE.DAT ze zmienną plikową „RF” }
  Reset(RF); { Otwarcie pliku rekordowego skojarzonego ze zmienną plikową „RF” }
  while not EOF(RF) do
  begin
{ Pętla wykonywana jest do momentu osiągnięcia końca pliku.
 O tym, że plik osiągnął swój koniec informuje funkcja EOF, która jest skojarzona ze zmienną
plikową „TF”. }

    Read(RF, Dane); { Wczytanie danych do obiektu „Dane” }

    Writeln(Dane.Nazwisko, ' ', Dane.Imie, ' ', Dane.Wiek); { Wyświetlenie zawartości obiektu
„Dane” }
  end;
  Close(RF); { Zamknięcie pliku skojarzonego ze zmienną plikową „RF” }
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.15. Przekazanie tablicy jako parametru do funkcji

Napisz program, który umożliwi wypełnienie danymi tablicy i przekazanie jej jako parametru do funkcji oraz wyświetli zawartość tej tablicy.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW26.PAS**.

```
uses Crt;

type
  TTablica = array[0..3] of Integer; { Deklaracja typu tablicowego „TTablica” }

procedure Wyswietl(Tablica :TTablica);
{ Procedura wyświetla zawartość tablicy }
var
  A :Integer;
begin
  for A:= 0 to 3 do
    Write(Tablica[A], ' ');
end;



var
  Tabliczka :TTablica; { Deklaracja zmiennej tablicowej }
begin
```

```
{ Wypełnienie tablicy danymi }
  Tabliczka[0]:= 11;
  Tabliczka[1]:= 12;
  Tabliczka[2]:= 13;
  Tabliczka[3]:= 14;

  ClrScr; { Czyści ekran }
  Writeln;
  Wyswietl(Tabliczka); { Przesłanie tablicy, jako parametru do funkcji „Wyswietl” }
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.16. Przeszukiwanie tablicy

Napisz program, który wypełni tablicę losowymi liczbami oraz umożliwi przeszukanie tej tablicy, w celu wyświetlenia informacji o ilości wystąpień szukanej liczby. Liczba ta ma być wprowadzona z klawiatury.

Sposób wykonania

Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW26A.PAS**.

```
uses Crt;

var
   Tabliczka :array[0..11] of Shortint; { Deklaracja zmiennej tablicowej }
  TT, A, Sum :Shortint; { Deklaracja zmiennych liczbowych całkowitych }
begin
  ClrScr;
  Writeln;
  Writeln('Tabela - wyszukiwanie wpisanej liczby');
  Writeln;

  Randomize; { Zainicjowanie generatora liczb losowych  }
  for TT:= 0 to 11 do
    Tabliczka[TT]:= Random(10); { Wypełnienie tablicy wylosowanymi liczbami }


{ Wyświetlenie zawartości tablicy }
  Writeln('Wylosowane liczby:');
  for TT:= 0 to 11 do
    Write(Tabliczka[TT], ' ');


  Writeln;
  Writeln;
  Write('Podaj liczbę:');
  Readln(A);
  Sum:= 0;
```

```
{ Liczenie wystąpień w tablicy podanej liczby }
  for TT:= 0 to 11 do
    if (Tabliczka[TT] = A) then Sum:= Sum+1;

{ Wyświetlenie informacji o ilości wystąpień w tablicy podanej z klawiatury liczby }
  Writeln;
  Writeln('Liczba ', A, ' powtarza się ', Sum, ' raz(y)');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.17. Rysowanie okien w trybie tekstowym

Napisz program, który narysuje dwa okna.

Sposób wykonania

Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW27.PAS**.

```
uses Crt;

function fChar(numSpc :Shortint; numChar :Byte) :String;
{ Funkcja zwraca łańcuch składający się ze znaku o numerze podanym w parametrze
„numChar" }
var
    TT :Shortint; { Deklaracja zmiennej liczbowej całkowitej }
  txtS :String; { Deklaracja zmiennej tekstowej }
begin
  { fChar }
  txtS:= ''; { Wyczyszczenie zmiennej tekstowej }

  for TT:= 0 to numSpc-1 do
    txtS:= txtS+CHR(numChar);
{
 "txtS:= txtS+CHR(numChar)" – dodanie do zmiennej „txtS" znaku, który jest otrzymany
dzięki funkcji „CHR()". CHR(X) – wyświetla znak o numerze podanym w parametrze „X".
Ilość znaku, jaka znajdzie się w zmiennej „txtS" zależy od liczby przechowywanej w zmiennej
liczbowej „numSpc".
}
  fChar:= txtS; { Funkcja zwraca ciąg tych samych znaków }
end;

procedure OknoRamka(wX, wY, wW, wH :Shortint; txtStr :String);
{ Procedura rysuje okno z jedną ramką }
var
  TT :Shortint; { Deklaracja zmiennej liczbowej całkowitej }
begin
  TextBackGround(BLUE); { Ustawienie czarnego koloru dla tła}
  TextColor(WHITE); { Ustawienie białego koloru dla tekstu }
```

```
  GotoXY(wX, wY); { Ustawienie kursora w pozycji wX, wY }
  Write(#218, fChar(wW-2, 196), #191); { Wyświetlenie górnej ramki okna }

{ Wyświetlenie pionowych ramek okna }
  for TT:= 0 to wH do
  begin
    GotoXY(wX, wY+1+TT);
    Write(#179, fChar(wW-2, 32), #179);
  end;

  GotoXY(wX, wY+wH+1);
  Write(#192, fChar(wW-2, 196), #217);

{ Wyświetlenie nazwy okna }
  if (txtStr<>'') then
  begin
{ Jeżeli parametr „txtStr" zawiera przynajmniej jeden znak, to wyświetl nazwę okna }

    GotoXY(wX+2, wY);
    Write(#32, txtStr, #32);
  end;
end;

procedure OknoRamki(wX, wY, wW, wH :Shortint; txtStr :String);
{ Procedura rysuje okno z dwoma ramkami }
var
  TT :Shortint;
begin
  TextBackGround(GREEN);
  TextColor(WHITE);
  GotoXY(wX, wY);
  Write(#201, fChar(wW-2, 205), #187);
  for TT:= 0 to wH do
  begin
    GotoXY(wX, wY+1+TT);
    Write(#186, fChar(wW-2, 32), #186);
  end;
  GotoXY(wX, wY+wH+1);
  Write(#200, fChar(wW-2, 205), #188);
  if (txtStr<>'') then
  begin
    GotoXY(wX+2, wY);
    Write(#32, txtStr, #32);
  end;
end;



begin
  TextBackGround(BLACK);
```

```
  TextColor(LightGRAY);
  ClrScr;
  Writeln;
  Writeln('== Okna ==');
  OknoRamka(5, 5, 30, 6, 'Okno 1');
  OknoRamki(40, 7, 30, 9, 'Okno 2');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.18. Kod iteracyjny i rekurencyjny

Napisz programy liczenia silni. Pierwszy program ma wykorzystywać kod iteracyjny, a drugi kod rekurencyjny.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW28_ir.PAS**.

Algorytm iteracyjny polega na powtarzaniu pewnych operacji wykonywanych przez instrukcje, funkcje lub procedury, w celu wykonania określonego zadania.

Algorytm rekurencyjny polega na wywoływaniu funkcji lub procedury przez samą siebie, w celu wykonania określonej operacji.

Poniżej przedstawione są oba algorytmy, prezentujące obliczenie silni:

```
uses Crt;

{ Algorytm iteracyjny }
function ITER_Silnia(N :Integer) :Longint;
var
  TT :Integer; { Deklaracja zmiennej liczbowej całkowitej }
  SP :Longint;
begin
  ITER_Silnia:= -1;

  SP:= 0; { Wyzerowanie zmiennej „SP” }

  if (N > -1) then
  begin
{ Jeżeli zawartość zmiennej „N” jest większa od –1, to wykonaj poniższe instrukcje }

    SP:= 1; { Przypisanie zmiennej „SP” liczby 1 }

    for TT:= 1 to N do
      Sp:= SP*TT; { Zwiększenie zmiennej „Sp” dzięki pomnożeniu jej przez wartość
znajdującą się w zmiennej „TT” }

    ITER_Silnia:= Sp; { Funkcja zwraca wynik przechowywany w zmiennej „Sp” }
```

```
  end;
end;

{ Algorytm rekurencyjny }
function REK_Silnia(N :Integer) :Longint;
begin
  if (N < 0) then
    REK_Silnia:= -1
  else
  if (N = 0) then
    REK_Silnia:= 1
  else
    REK_Silnia:= N*REK_Silnia(N-1); { REK_Silnia(N-1) – wywołanie instrukcji przez samą
siebie, z jednoczesnym pomniejszeniem zmiennej „N” o wartość 1 }
end;



begin
  ClrScr;
  Writeln('== Kod iteracyjny i rekurencyjny ==');
  Writeln;
  Writeln('Algorytm iteracyjny (Silnia z 10): ', ITER_Silnia(10));
  Writeln;
  Writeln('Algorytm rekurencyjny (Silnia z 10): ', REK_Silnia(10));
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.19. Obliczenie wieku

Napisz program, który poda twój wiek, na podstawie wprowadzonej daty urodzenia z klawiatury.

Sposób wykonania

Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW29.PAS**.

```
uses Crt, Dos;
{
 Po słowie USES następuje deklaracja modułów, których instrukcje są wykorzystywane w
aktualnie pisanym programie.
Moduł CRT jest odpowiedzialny za obsługę ekranu i klawiatury.
Moduł DOS jest odpowiedzialny za wykonywanie funkcji i obsługę systemu operacyjnego.
}

function IntToStr(X :Longint) :String;
{ Funkcja przekształca liczbę całkowitą na wartość tekstową liczbową (ciąg znaków) }
var
  txtText :String; { Deklaracja zmiennej tekstowej }
```

```
begin
  { IntToStr }
  STR(X, txtText);
  IntToStr:= txtText; { Funkcja zwraca wartość tekstową }
end;

function StrToInt(txtStr :String) :Longint;
{ Funkcja przekształca ciąg cyfr na wartość liczbową, typu całkowitego }
var
   numNumber :Longint; { Deklaracja zmiennej liczbowej, typu całkowitego }
  numErrCode :Integer; { Deklaracja zmiennej liczbowej, typu całkowitego }
begin
  { StrToInt }
  VAL(txtStr, numNumber, numErrCode);
  StrToInt:= numNumber; { Funkcja zwraca wartość numeryczną, typu całkowitego }
end;

function DodajZeroPrzed(txtStr :String) :String;
{
Funkcja dodaje zero przed pojedynczą liczbą traktowaną jako ciąg znaków, w innym
przypadku zwraca dwie liczby.

Przykład:
  1) podana została liczba 10, to funkcja zwróci nam liczbę 10 jako ciąg dwóch znaków
  2) podana została liczba 3, to funkcja zwróci nam liczbę 03 jako ciąg dwóch znaków
}
begin
  { DodajZeroPrzed }

{ Funkcja zwraca liczbę zawartą w parametrze „txtStr” }
  DodajZeroPrzed:= txtStr;

  if (Length(txtStr) = 1) then DodajZeroPrzed:= '0'+txtStr;
   {
Jeżeli ilość znaków w zmiennej "txtStr" jest równa jeden, to wykonaj instrukcję po słowie
THEN (tj. dodaj przed cyfrą zero).

 ================================================

    Length(TEKST) - Funkcja zwraca liczbę znaków zawartą w parametrze TEKST.
   }
end;

function DATA_DzisJest :String;
{ Funkcja podaje bieżącą datę pobraną z komputera }
var
  YE, M, D, DOW :Word;
begin
  GetDate(YE, M, D, DOW); { Pobiera bieżącą datę systemową i dzieli ją na rok (YE),
miesiąc (M), dzień(D) i dzień tygodnia(DOW) }
```

```
  DATA_DzisJest:= IntToStr(YE)+'-'+
        DodajZeroPrzed(IntToStr(M))+'-'+
        DodajZeroPrzed(IntToStr(D)); { Zwraca datę w formacie RRRR-MM-DD }
end;




var
  txtDataUr :String; { Deklaracja zmiennej tekstowej }
     Wiek :Integer; { Deklaracja zmiennej liczbowej całkowitej }
begin
  ClrScr; { Czyści ekranu }
  Writeln; { Przejście o jeden wiersz w dół }
  Writeln('== Obliczanie lat ==');
  Writeln('Dzisiaj jest:'+CHR(32)+DATA_DzisJest);
  Writeln;
  Write('Podaj datę urodzenia:');
  Readln(txtDataUr); { Pobiera datę wpisaną z klawiatury i przypisuje ją do zmiennej
„txtDataUr" }

  Wiek:= 0;
  Wiek:= StrToInt(Copy(DATA_DzisJest, 1, 4))-StrToInt(txtDataUr);
   {
        Copy(Ciag_Znakow, Pozycja_Pierwszego_Znaku, Liczba_Znakow)

        Funkcja zwraca fragment ciągu znaków, o odpowiedniej ilości
        znaków określonych w parametrze "Liczba_Znakow", od
        pozycji "Pozycja_Pierwszego_Znaku".
        UWAGA: Znak spacji traktowany jest jako znak.

        Na przykład: Copy('To jest tekst', 4, 6) - efektem pracy tej
        funkcji będzie wyciągnięcie fragmentu tekstu "jest t".
   }

  Writeln;
  Writeln('Masz '+IntToStr(Wiek)+' lat(a)');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 4.20. Separatory tekstu

Napisz program, umożliwiający podział tekstu na kilka wierszy. Wykorzystaj do podzielenia tekstu separatorów tekstu.

Przykład

Jest podany tekst „ATARI;SPECTRUM;AMIGA;ATARI TT”, separatorem tekstu jest znak średnika. Należy napisać taką funkcję, która każdy wyraz umieści w kolejnych wierszach, tj.
Atari
Spectrum
Amiga
Atari TT

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW30.PAS**.

```
uses Crt;

function TEXT_jbGetColumnText(txtFind :String; var txtString :String) :String;
{ Funkcja umożliwia podział łańcucha znaków, wykorzystując separator tekstu }
var
  numLen, numPos :Integer; { Deklaracja zmiennych liczbowych całkowitych }
       txtTemp :String; { Deklaracja zmiennej tekstowej }
begin
  { TEXT_jbGetColumnText }
  TEXT_jbGetColumnText:= ''; { Funkcja zwraca znak pusty }

  if (txtFind<>'') and (txtString<>'') then
  begin
{ Jeżeli zmienna „txtFind” i zmienna „txtString” przechowują przynajmniej jeden znak, to
wykonaj poniższe instrukcje }

    numLen:= 0; { Wyzerowanie zmiennej liczbowej }
    numLen:= Length(txtFind); { Przypisanie do zmiennej liczbowej liczby określającej ilość
znaków w tekście, który przechowywany jest w zmiennej „txtFind” }

    numPos:= 0;
    numPos:= Pos(txtFind, txtString);
    {
      Pos(Szukane_Znaki, Lancuch_Znakow)

      Funkcja zwraca pozycję występowania pierwszego znaku podłańcucha (parametr
"Szukane_Znaki") w ciągu znaków (parametr "Lancuch_Znakow").
    }

    TEXT_jbGetColumnText:= txtString; { Funkcja zwraca tekst, który przechowywany jest w
zmiennej „txtString” }

    if (numPos<>0) then
    begin
{ Jeżeli został odnaleziony znak w tekście (tj. Zmienna „numPos” jest różna od zera), to
wykonaj poniższe instrukcje }

      TEXT_jbGetColumnText:= Copy(txtString, 1, numPos-1); { Funkcja zwraca okrojony
przez funkcję COPY tekst znajdujący się w zmiennej „txtString”.
          Copy(Ciag_Znakow, Pozycja_Pierwszego_Znaku, Liczba_Znakow)
```

```
        Funkcja zwraca fragment ciągu znaków, o odpowiedniej ilości
        znaków określonych w parametrze "Liczba_Znakow", od
        pozycji "Pozycja_Pierwszego_Znaku".
        UWAGA: Znak spacji traktowany jest jako znak.

        Na przykład: Copy('To jest tekst', 4, 6) - efektem pracy tej
      funkcji będzie wyciągnięcie fragmentu tekstu "jest t".
}

    txtTemp:= '';
    txtTemp:= Copy(txtString, numPos+numLen, Length(txtString));

    txtString:= '';
    txtString:= txtTemp;
   end;

  end;
end;



var
    txtTekst :String;
  numZnak, TT :Integer;
begin
  txtTekst:= 'Atari;Spectrum;Amiga;Atari TT';


  numZnak:= 0;

{ Oblicza ilość separatorów tekstu }
  for TT:= 1 to Length(txtTekst) do
    if (Copy(txtTekst, TT, 1)=';') then numZnak:= numZnak+1;


  ClrScr;
  Writeln;
  Writeln('== Podzielenie tekstu na kilka wierszy ==');

{ Dzieli łańcuch znaków }
  for TT:= 0 to numZnak do
    Writeln(TEXT_jbGetColumnText(';', txtTekst));
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

5. Ćwiczenia zaawansowane

W rozdziale tym przedstawione są ćwiczenia, mające na celu zaprezentowanie gotowych praktycznych programów.

Ćwiczenie 5.1. Przypominacz

Napisz program, który będzie przypominał o ważnych datach (np. herbatka u cioci), ponadto program ma uwzględniać ostatni dzień miesiąca lub umożliwić wyświetlanie informacji co miesiąc. Dane mają być umieszczone w pliku tekstowym.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW31Z.PAS**.

```
uses Crt, Dos;

function FileExists(txtFileName :String) :Boolean;
{ Funkcja sprawdza, czy na dysku istnieje fizycznie plik o nazwie podanej w parametrze
„txtFileName”. Jeżeli plik istnieje, to funkcja zwróci wartość TRUE (Prawda), w przeciwnym
przypadku funkcja zwróci FALSE (Fałsz) }
var
  Info :SearchRec;
begin
  { FileExists }
  FindFirst(txtFileName, Directory, Info);
  FileExists:= FALSE;
  if (DosError = 0) then FileExists:= TRUE;
end;

function IntToStr(X :Longint) :String;
{ Funkcja przekształca wartość numeryczną na postać tekstową liczbową }
var
  txtText :String;
begin
  { IntToStr }
  STR(X, txtText);
  IntToStr:= txtText;
end;

function StrToInt(txtStr :String) :Longint;
{ Funkcja przekształca postać tekstową liczbową wartość numeryczną }
var
   numNumber :Longint;
```

```
  numErrCode :Integer;
begin
  { StrToInt }
  VAL(txtStr, numNumber, numErrCode);
  StrToInt:= numNumber;
end;

function DodajZeroPrzed(txtStr :String) :String;
{ Funkcja dodaje cyfrę „0” przed pojedynczą cyfrą, np. 1 -> 01 }
begin
  { DodajZeroPrzed }
  DodajZeroPrzed:= txtStr;
  if (Length(txtStr) = 1) then DodajZeroPrzed:= '0'+txtStr;
end;

function DATA_DzisJest :String;
{ Funkcja podaje datę bieżącą, pobraną z komputera }
var
  YE, M, D, DOW :Word;
begin
  GetDate(YE, M, D, DOW);
  DATA_DzisJest:= IntToStr(YE)+'-'+
          DodajZeroPrzed(IntToStr(M))+'-'+
          DodajZeroPrzed(IntToStr(D));
end;

function DATA_DzisJestDzien :String;
{ Funkcja podaje nazwę dnia, który jest określany na podstawie bieżącej daty systemowej }
const
  DzienTygodnia :array [0..6] of String = ('Niedziela',
                                           'Poniedzialek',
                                           'Wtorek',
                                           'Sroda',
                                           'Czwartek',
                                           'Piatek',
                                           'Sobota');
var
  YE, M, D, DOW :Word;
begin
  GetDate(YE, M, D, DOW);
  DATA_DzisJestDzien:= DzienTygodnia[DOW];
end;

function DATA_MiesiacMaDni(txtDate :String) :Shortint;
{ Funkcja podaje ilość dni, jaką posiada miesiąc, na podstawie podanej daty }
var
  numMonth, numMonthToDays :Shortint;
            numYear :Integer;
begin
  { DATA_MiesiacMaDni }
```

```
  numYear:= 0;
  numYear:= StrToInt(Copy(txtDate, 1, 4));

  numMonth:= 0;
  numMonth:= StrToInt(Copy(txtDate, 6, 2));

  numMonthToDays:= 0;
  if (numMonth in [1, 3, 5, 7, 8, 10, 12]) then numMonthToDays:= 31;
  if (numMonth in [4, 6, 9, 11]) then numMonthToDays:= 30;
  if (numMonth = 2) then
  begin
    numMonthToDays:= 28;
    if ((numYear mod 4 = 0) and (numYear mod 100 <>0)) or
       (numYear mod 400 = 0) then
    begin
      numMonthToDays:= 0;
      numMonthToDays:= 29;
    end;
  end;
  DATA_MiesiacMaDni:= numMonthToDays;
end;

function jtbPrzypominacz(txtDataMD :String) :Boolean;
{ Funkcja sprawdza, czy podana data zgadza się z datą pobraną z komputera. Jeżeli daty się
zgadzają to funkcja zwraca wartość TRUE (Prawda), w innym przypadku funkcja zwraca
FALSE (Fałsz) }
var
  txtRMiesiac, txtRDzien, txtNazwaDnia :String;
               txtMiesiac, txtDzien :String;
                     numIloscDni :Shortint;
begin
  { jtbPrzypominacz }
  jtbPrzypominacz:= FALSE;

    txtRMiesiac:= '';
    txtRMiesiac:= Copy(DATA_DzisJest, 6, 2);

    txtRDzien:= '';
    txtRDzien:= Copy(DATA_DzisJest, 9, 2);

    txtNazwaDnia:= '';
    txtNazwaDnia:= Copy(DATA_DzisJestDzien , 1, 2);

    numIloscDni:= 0;
    numIloscDni:= DATA_MiesiacMaDni(DATA_DzisJest);

  if (txtDataMD<>'') then
  begin

    txtMiesiac:= '';
```

```
   txtMiesiac:= DodajZeroPrzed(Copy(txtDataMD, 1, 2));

   txtDzien:= '';
   txtDzien:= DodajZeroPrzed(Copy(txtDataMD, 4, 2));
   if (txtDzien = '!!') then
   begin
    txtDzien:= '';
    txtDzien:= IntToStr(numIloscDni);
   end;

   if (txtMiesiac+'-'+txtDzien = txtRMiesiac+'-'+txtRDzien) or
      (txtMiesiac+'-'+txtDzien = txtRMiesiac+'---') or
      (txtMiesiac+'-'+txtDzien = '---'+txtRDzien) or
      (txtMiesiac+'-'+txtDzien = txtRMiesiac+'-'+txtNazwaDnia) or
      (txtMiesiac+'-'+txtDzien = '---'+txtNazwaDnia) or
      (txtMiesiac+'-'+txtDzien = '-----') then jtbPrzypominacz:= TRUE;
  end;
end;

procedure OdczytajPlik;
{ Procedura odczytuje plik tekstowy, który zawiera daty poszczególnych uroczystości lub
spraw do załatwienia }
const
  txtNazwaPliku = 'reminder.txt';
var
  txtCzytaj :String;
        FT :Text;
        TT :Longint;
     okJest :Boolean;
begin
  { Przypominacz }
  if (FileExists(txtNazwaPliku) = TRUE) then
  begin
    Assign(FT, txtNazwaPliku);
    Reset(FT);
    TT:= 0;
    while not EOF(FT) do
    begin
      txtCzytaj:= '';
      ReadLn(FT, txtCzytaj);

      okJest:= FALSE;
      okJest:= jtbPrzypominacz(Copy(txtCzytaj, 1, 5));
      if (okJest = TRUE) then
      begin
        Writeln(IntToStr(TT+1)+')'+CHR(32)+
              Copy(DATA_DzisJest, 6, 5)+CHR(32)+
              Copy(txtCzytaj, 7, 77));
        TT:= TT+1;
      end;
```

```
    end;
    Close(FT)
  end;

  { Nacisnij dowolny klawisz... }
  Writeln;
  if (TT >0) then
  begin
    Write('Nacisnij dowolny klawisz...');
    repeat
    until KeyPressed;
  end;
end;



begin
  Writeln;
  Writeln('== Przypominacz ==');
  Writeln('Copyright(c)by Jan-Tadeusz Biernat');
  Writeln;
  Writeln('Dzisiaj jest:'+CHR(32)+DATA_DzisJest);
  Writeln('Dzień:'+CHR(32)+DATA_DzisJestDzien);
  Writeln;
  OdczytajPlik;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 5.2. Słownik

Napisz program SŁOWNIK, który umożliwi wyświetlenie znaczenia podanego słówka. Słówko, którego znaczenia chcemy się dowiedzieć, należy podać jako parametr przy uruchomieniu programu.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW32Z.PAS**.

```
uses Crt, Dos;

function LITERY_Male(txtStr :String) :String;
{ Funkcja zamienia duże litery na małe }

  function MalyZnak(chZnak :Char) :Char;
  { Funkcja zamienia dużą literę na małą literę }
  var
    TT :Byte;
  begin
    for TT:= 65 to 90 do
```

```
    if (Chr(TT) = chZnak) then chZnak:= Chr(TT+32);
   MalyZnak:= chZnak;
 end;

var
 I :Byte;
begin
 for I:= 1 to Length(txtStr) do
  txtStr[I]:= MalyZnak(txtStr[I]);
 LITERY_Male:= txtStr;
end;

function LITERY_Duze(txtStr :String) :String;
{ Funkcja zamienia małe litery na duże }

 function DuzyZnak(chZnak :Char) :Char;
 { Funkcja zamienia małą literę na dużą literę }
 var
  TT :Byte;
 begin
  for TT:= 97 to 122 do
    if (Chr(TT) = chZnak) then chZnak:= Chr(TT-32);
  DuzyZnak:= chZnak;
 end;

var
 I :Byte;
begin
 for I:= 1 to Length(txtStr) do
  txtStr[I]:= DuzyZnak(txtStr[I]);
 LITERY_Duze:= txtStr;
end;

function FileExists(txtFileName :String) :Boolean;
{ Funkcja sprawdza, czy plik o podanej nazwie fizycznie znajduje się na dysku. Jeżeli tak, to
funkcja zwróci wartość TRUE (Prawda). W innym przypadku funkcja zwróci wartość FALSE
(Fałsz) }
var
 Info :SearchRec;
begin
 { FileExists }
 FindFirst(txtFileName, Directory, Info);
 FileExists:= FALSE;
 if (DosError = 0) then FileExists:= TRUE;
end;

function Slownik(txtZnajdz :String) :Boolean;
{ Funkcja odczytuje plik tekstowy i wyszukuje znaczenie słówka podanego w parametrze
„txtZnajdz". Gdy znaczenie danego słówka zostanie odnalezione, to następuje jego
wyświetlenie }
```

```
const
  txtNazwaPliku = 'slow.txt';
var
  txtSzukaj, txtCzytaj :String;
                FT :Text;
begin
  if (FileExists(txtNazwaPliku) = TRUE) then
  begin
    Assign(FT, txtNazwaPliku);
    Reset(FT);
    while not EOF(FT) do
    begin
      txtSzukaj:= '';
      ReadLn(FT, txtSzukaj);
      if (LITERY_Male(txtSzukaj) = '['+LITERY_Male(txtZnajdz)+']') then
      begin
        Writeln(LITERY_Duze(txtZnajdz));
        while not EOF(FT) do
        begin
          txtCzytaj:= '';
          ReadLn(FT, txtCzytaj);
          if (Copy(txtCzytaj, 1, 1) = '[') then Break;
          Writeln(txtCzytaj);
        end;
      end;
    end;
    Close(FT)
  end;
end;



begin
  ClrScr;
  Writeln;
  Writeln('== Słownik ==');
  Writeln;
  if (ParamStr(1)<>'') then
    Slownik(ParamStr(1))
  else
    Writeln('Musisz podac wyraz jako parametr!');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Po skompilowaniu pliku o nazwie **CW32Z.PAS** otrzymujemy plik **CW32Z.EXE**. Wykorzystując polecenie menu **Tryb MS-DOS**, w linii poleceń musisz wpisać nazwę programu i słówko, którego znaczenia chcesz się dowiedzieć.

Przykład: CW32Z.EXE dysk (słowo „dysk” jest parametrem, którego znaczenia chcesz się dowiedzieć) i po naciśnięciu klawisza ENTER, zostanie wyświetlone znaczenie słówka „Dysk”.

Ćwiczenie 5.3. Menu

Napisz program ilustrujący zasadę działania menu. Oprócz możliwości wyboru jednego z kilku elementów, ma również umożliwić opuszczenie programu po naciśnięciu klawisza ENTER lub ESC z jednoczesnym wyświetleniem wybranego elementu.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW33Z.PAS**.

```
uses Crt;

function fSpc(numSpc :Shortint) :String;
{ Funkcja zwraca ciąg znaków pustych. Ilość tych znaków zależy od liczby przechowywanej w
parametrze „numSpc” }
var
   TT :Shortint;
  txtS :String;
begin
  { fSpc }
  txtS:= '';
  for TT:= 0 to numSpc-1 do
    txtS:= txtS+CHR(32);
  fSpc:= txtS;
end;

procedure Menu(mX, mY, mS :Shortint; txtStr :String;
            numMenu :Shortint; var numAct :Shortint);
{ Procedura wyświetla element menu.
Parametr „numMenu” określa numer kolejno wprowadzonego elementu. Natomiast parametr
„numAct” umożliwia zaznaczenie elementu, poprzez podanie jego numeru. Zaznaczenie
elementu w menu następuje dopiero po porównaniu numerów przechowywanych w
parametrach „numMenu” i „numAct”. Jeżeli oba numery są takie same, to następuje
podświetlenie wybranego elementu o numerze przechowywanym w parametrze „numMenu” }
begin
  { Menu }
  TextBackGround(BLACK);
  TextColor(WHITE);
  GotoXY(mX, mY);
  Write(fSpc(mS+2));
  GotoXY(mX+2, mY);
  Write(Copy(txtStr, 1, mS-2));
  if (numMenu = numAct) then
  begin
    TextBackGround(WHITE);
    TextColor(BLACK);
```

```
    GotoXY(mX, mY);
    Write(fSpc(mS+2));
    GotoXY(mX+2, mY);
    Write(Copy(txtStr, 1, mS-2));
  end;
end;

function KN_KlawiszNacisnij :Integer;
{ Funkcja podaje kod (tj. numer) naciśniętego klawisza na klawiaturze }
var
  numKey :Integer;
     Ch :Char;
begin
  { KN_KlawiszNacisnij }
  repeat until KeyPressed;
    Ch:= ReadKey;
    numKey:= Ord(Ch);
    if (Ch = #0) then
    begin
      Ch:= ReadKey; { trap function keys: 359 - Help; 9 - TAB; 368 - F10 }
      numKey:= Ord(Ch)+300;
    end;

  {
  Funkcja „KeyPressed" służy do sprawdzenia, czy nastąpiło naciśnięcie na klawiaturze
dowolnego klawisza. Jeżeli klawisz został naciśnięty, to funkcja „KeyPressed" daje wartość
TRUE. W przeciwnym przypadku wartością funkcji jest FALSE.

Jeżeli wartością funkcji „KeyPressed" jest TRUE, to ponowne ustawienie tej funkcji na
wartość FALSE następuje poprzez użycie funkcji „ReadKey".

Pętla REPEAT…UNTIL będzie wykonywana tak długo, dopóki nie zostanie naciśnięty klawisz
na klawiaturze. Sprawdzenie, czy klawisz na klawiaturze został naciśnięty jest wykonywane na
końcu pętli REPEAT…UNTIL.

Poniższa konstrukcja:
    if (Ch = #0) then
    begin
      Ch:= ReadKey; { trap function keys: 359 - Help; 9 - TAB; 368 - F10 }
      numKey:= Ord(Ch)+300;
    end;
umożliwia wykrycie naciśniętego klawisza funkcyjnego lub nawigacyjnego (tzw. kursory).
Dzięki temu do naciśniętego klawisza zostaje dodanie wartość 300, co umożliwia
oprogramowanie klawiszy funkcyjnych i nawigacyjnych.
  }


  KN_KlawiszNacisnij:= numKEY; { Funkcja zwraca numer naciśniętego klawisza }
end;
```

```
procedure KlawiszeMenu;
{ Procedura buduje i obsługuje menu }
var
  numMenu :Shortint;
   numKey :Integer;
begin
  { KlawiszeMenu }
  numMenu:= 0;
  numMenu:= 1;
  repeat
    Menu(5, 4, 12, 'Element 1', 1, numMenu); { Dodanie nowego elementu menu }
    Menu(5, 5, 12, 'Element 2', 2, numMenu);
    Menu(5, 6, 12, 'Element 3', 3, numMenu);
    Menu(5, 7, 12, 'Element 4', 4, numMenu);
    Menu(5, 8, 12, 'Element 5', 5, numMenu);
    Menu(5, 9, 12, 'Element 6', 6, numMenu);
    Menu(5, 10, 12, 'Element 7', 7, numMenu);
    Menu(5, 11, 12, 'Element 8', 8, numMenu);
    Menu(5, 12, 12, 'Element 9', 9, numMenu);
    numKey:= 0;
    numKey:= KN_KlawiszNacisnij;
    if (numKey = 372) then Dec(numMenu); { Up }
    if (numKey = 380) then Inc(numMenu); { Down }

    if (numMenu > 9) then numMenu:= 1;
    if (numMenu < 1) then numMenu:= 9;
  until (numKey = 27) or (numKey = 13);
  TextBackGround(BLACK);
  TextColor(LightGRAY);
  GotoXY(5, 15);
  Writeln('Wybrales: Element ', numMenu);
end;



begin
  TextBackGround(BLACK);
  TextColor(LightGRAY);
  ClrScr;
  Writeln;
  Writeln('== Menu ==');
  KlawiszeMenu;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 5.4. Wprowadzanie danych – obsługa klawiatury

Napisz program, który umożliwi wprowadzanie danych z klawiatury. Program ma również umożliwić przerwanie wprowadzania danych, przez naciśnięcie klawisza ESC.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW34Z.PAS**.

```
uses Crt;

procedure KT_KursorTekst(X, Y, TS :Shortint; txtStr :String);
{ Procedura wyświetla wprowadzony z klawiatury tekst }
var
     TT :Shortint;
  txtSpc :String;
begin
  { KT_KursorTekst }
  txtSpc:= '';
  for TT:= 0 to TS do
    txtSpc:= txtSpc+CHR(32);

  TextBackGround(BLACK);
  GotoXY(X, Y);
  Write(txtSpc);
  TextColor(WHITE);
  GotoXY(X, Y);
  Write(txtStr);
end;

function KN_KlawiszNacisnij :Integer;
{ Funkcja podaje kod (tj. numer) naciśniętego klawisza na klawiaturze }
var
  numKey :Integer;
     Ch :Char;
begin
  { KN_KlawiszNacisnij }
  repeat until KeyPressed;
    Ch:= ReadKey;
    numKey:= Ord(Ch);
    if (Ch = #0) then
    begin
      Ch:= ReadKey; { trap function keys: 359 - Help; 9 - TAB; 368 - F10 }
      numKey:= Ord(Ch)+300;
    end;
  KN_KlawiszNacisnij:= numKEY; { Funkcja zwraca numer naciśniętego klawisza }
end;


function Klawisze(X, Y, TS :Shortint; txtStr :String) :String;
{ Funkcja obsługuje klawiaturę, poprzez kontrolę ilości wprowadzanych znaków. ESC –
umożliwia rezygnację z wprowadzania znaków; ENTER – zatwierdza wprowadzony tekst }
```

```
var
  txtTekst :String;
    numLen :Shortint;
    numKey :Integer;
begin
  { Klawisze }
  TextBackGround(BLACK);
  TextColor(WHITE);
  GotoXY(X, Y);
  Write(txtStr+':');
  txtTekst:= '';
  repeat
    KT_KursorTekst(X+Length(txtStr)+1, Y, TS, txtTekst);
    numKey:= 0;
    numKey:= KN_KlawiszNacisnij;
    if (numKey = 8) then
    begin
      numLen:= 0;
      numLen:= Length(txtTekst);
      txtTekst:= Copy(txtTekst, 1, Length(txtTekst)-1);
    end
    else
    if (numKey = 27) then
    begin
      Klawisze:= '';
    end
    else
    if (numKey = 13) then
    begin
      Klawisze:= txtTekst;
    end
    else
    begin
      if (Length(txtTekst) < TS) then
      begin
        txtTekst:= txtTekst+CHR(numKey);
      end;
    end;
  until (numKey = 27) or (numKey = 13);
  TextBackGround(BLACK);
  TextColor(LightGRAY);
end;
```

```
var
  txtNapis :String;
begin
  TextBackGround(BLACK);
  TextColor(LightGRAY);
```

```
  ClrScr;
  Writeln;
  Writeln('== Obsługa klawiatury ==');
  txtNapis:= '';
  txtNapis:= Klawisze(4, 4, 44, 'Wprowadz tekst'); { Wywołanie funkcji „Klawisze” obsługującej klawiaturę }

  GotoXY(4, 6);
  Write('Wprowadzony tekst to: '+txtNapis);
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 5.5. Tabliczka mnożenia

Napisz program, który umożliwi sprawdzenie znajomości tabliczki mnożenia.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa programu to **CW35Z.PAS**.

```
uses Crt;

procedure KT_KursorTekst(X, Y, TS :Shortint; txtStr :String);
{ Procedura wyświetla tekst wprowadzony z klawiatury }
var
     TT :Shortint;
  txtSpc :String;
begin
  { KT_KursorTekst }
  txtSpc:= '';
  for TT:= 0 to TS do
    txtSpc:= txtSpc+CHR(32);

  TextBackGround(BLACK);
  GotoXY(X, Y);
  Write(txtSpc);
  TextColor(WHITE);
  GotoXY(X, Y);
  Write(txtStr);
end;

function KN_KlawiszNacisnij :Integer;
{ Funkcja podaje kod (tj. numer) naciśniętego klawisza na klawiaturze }
var
  numKey :Integer;
     Ch :Char;
begin
  { KN_KlawiszNacisnij }
  repeat until KeyPressed;
```

```
    Ch:= ReadKey;
    numKey:= Ord(Ch);
    if (Ch = #0) then
    begin
      Ch:= ReadKey; { trap function keys: 359 - Help; 9 - TAB; 368 - F10 }
      numKey:= Ord(Ch)+300;
    end;
  KN_KlawiszNacisnij:= numKEY;
end;


function Klawisze(X, Y, TS :Shortint; txtStr :String) :String;
{ Obsługa klawiatury i weryfikacja wprowadzonych danych. Możliwe jest wprowadzenie tylko
cyfr. }
var
  txtTekst :String;
    numLen :Shortint;
    numKey :Integer;
begin
  { Klawisze }
  TextBackGround(BLACK);
  TextColor(WHITE);
  GotoXY(X, Y);
  Write(txtStr+':');
  txtTekst:= '';
  repeat
    KT_KursorTekst(X+Length(txtStr)+1, Y, TS, txtTekst);
    numKey:= 0;
    numKey:= KN_KlawiszNacisnij;
    if (numKey = 8) then
    begin
      numLen:= 0;
      numLen:= Length(txtTekst);
      txtTekst:= Copy(txtTekst, 1, Length(txtTekst)-1);
    end
    else
    if (numKey = 27) then
    begin
      Klawisze:= '';
    end
    else
    if (numKey = 13) then
    begin
      Klawisze:= txtTekst;
    end
    else
    begin
      if (Length(txtTekst) < TS) then
      begin
        if ((numKey > 47) and (numKey < 58)) then
```

```
        txtTekst:= txtTekst+CHR(numKey);
      { Umożliwia wprowadzenie tylko cyfr }
     end;
    end;
  until (numKey = 27) or (numKey = 13);
  TextBackGround(BLACK);
  TextColor(LightGRAY);
end;

procedure MT_Cyfry(X, Y, Cyfra :Shortint);
{ Procedura wyświetla liczby }
const
  txtCyfry1 :array[0..10] of String = (
    '  ####      ##      ###     ####       ## ',
    ' ######  ######    ######## #########     ### ',
    '##   ## ## ##    ##    ## ##    ##    #### ',
    '##    ##   ##    ##    #      ##    ## ## ',
    '##    ##   ##        ##    #####   ## ## ',
    '##    ##   ##       ###    #####  ##  ## ',
    '##    ##   ##       ##       ### ##   ## ',
    '##    ##   ##     ###          ## ######### ',
    '##   ##    ##    ##     #    ###     ## ',
    ' ######  ########## ######### #########     ##### ',
    '  ####   ########## #########   ####      ##### ');

  txtCyfry2 :array[0..10] of String = (
    '#######      #### #########   ####    ####   ',
    '#######    ######## ######### ######## ####### ',
    '#       ###     #    ## ##    ## ##    ## ',
    '#       ##          ## ##    # #    ## ',
    '######## ## ####      ##  ##  ### ##    ## ',
    '#    ## #########     ##  ######## ### #### ',
    '      ## ###   ##    ##  ###   ##  ##### # ',
    '      ## ##    #    ##  ##    #      ## ',
    '##    ### ##   ##   ##   ##   ##     ## ',
    '######### #######    ##   ######## #######  ',
    '  ####     ####     #     ####   ####    ');
var
  numKol :Shortint;
     TT :Shortint;
begin
  { MT_Cyfry }
  TextColor(LightBLUE);
  for TT:= 0 to 10 do
  begin
    if (Cyfra = 0) or (Cyfra = 5) then numKol:= 1;
    if (Cyfra = 1) or (Cyfra = 6) then numKol:= 11;
    if (Cyfra = 2) then numKol:= 22;
    if (Cyfra = 3) then numKol:= 33;
    if (Cyfra = 4) then numKol:= 43;
```

```
    if (Cyfra = 7) then numKol:= 21;
    if (Cyfra = 8) then numKol:= 31;
    if (Cyfra = 9) then numKol:= 42;
    GotoXY(X, Y+TT);
    if ((Cyfra >-1) and (Cyfra < 5)) then
      Write(Copy(txtCyfry1[TT], numKol, 11))
    else
    if ((Cyfra >4) and (Cyfra < 10)) then
      Write(Copy(txtCyfry2[TT], numKol, 11));
  end;
end;

procedure MT_ZnakRazy(X, Y: Shortint);
{ Procedura wyświetla znak mnożenia „ * ” }
const
  txtRazy :array[0..7] of String = (
    '#####  #### ',
    '#####  #### ',
    ' ###  ##  ',
    '   ####   ',
    '  #####   ',
    ' ##   ### ',
    '#####  #####',
    '#####  #####');
var
  TT :Shortint;
begin
  { MT_ZnakRazy }
  TextColor(LightBLUE);
  for TT:= 0 to 7 do
  begin
    GotoXY(X, Y+TT);
    Write(txtRazy[TT]);
  end;
end;

function IntToStr(X :Longint) :String;
{ Funkcja konwertuje liczbę na tekst }
var
  txtText :String;
begin
  { IntToStr }
  STR(X, txtText);
  IntToStr:= txtText;
end;

function StrToInt(txtStr :String) :Longint;
{ Funkcja konwertuje tekst na liczbę }
var
   numNumber :Longint;
```

```
  numErrCode :Integer;
begin
  { StrToInt }
  VAL(txtStr, numNumber, numErrCode);
  StrToInt:= numNumber;
end;

function fSpc(numSpc :Shortint) :String;
{ Funkcja zwraca łańcuch składający się z pustych znaków }
var
    TT :Shortint;
  txtS :String;
begin
  { fSpc }
  txtS:= '';
  for TT:= 0 to numSpc-1 do
    txtS:= txtS+CHR(32);
  fSpc:= txtS;
end;



var
  numLicz1, numLicz2, numWynik, TT :Shortint;
            txtLiczba, txtWynik :String;
begin
  TextBackGround(BLACK);
  TextColor(LightGRAY);
  ClrScr;
  Writeln('Tabliczka Mnożenia (c)by Jan-Tadeusz Biernat');
  repeat
    Randomize;
    numLicz1:= 0;
    numLicz1:= Random(10);

    numLicz2:= 0;
    numLicz2:= Random(10);

    numWynik:= 0;
    numWynik:= numLicz1*numLicz2;

    if (numLicz1 > 9) then
    begin
      txtLiczba:= '';
      txtLiczba:= IntToStr(numLicz1);
      MT_Cyfry(5, 5, StrToInt(Copy(txtLiczba, 1, 1)));
      MT_Cyfry(15, 5, StrToInt(Copy(txtLiczba, 2, 1)));
    end
    else
    begin
```

```
    MT_Cyfry(15, 5, numLicz1);
  end;

  MT_ZnakRazy(32, 8);

  if (numLicz2 > 9) then
  begin
    txtLiczba:= '';
    txtLiczba:= IntToStr(numLicz2);
    MT_Cyfry(48, 5, StrToInt(Copy(txtLiczba, 1, 1)));
    MT_Cyfry(59, 5, StrToInt(Copy(txtLiczba, 2, 1)));
  end
  else
  begin
    MT_Cyfry(48, 5, numLicz2);
  end;

  txtWynik:= '';
  txtWynik:= Klawisze(25, 19, 3, 'Twoja odpowiedz');

  if (txtWynik<>'') then
  begin
    GotoXY(47, 20);
    if (StrToInt(txtWynik) = numWynik) then
      Write('Bardzo dobrze!')
    else
      Write('Źle!');
    Delay(999);
    GotoXY(47, 20);
    Write(fSpc(14));
  end;

  { CLS }
  for TT:= 0 to 15 do
  begin
    GotoXY(4, 4+TT);
    Write(fSpc(67));
  end;
 until (txtWynik = '');
 TextBackGround(BLACK);
 TextColor(LightGRAY);
 ClrScr;
 Writeln('Pa pa');
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 5.6. Zamiana jednego ciągu znaków na inny ciąg znaków w pliku

Napisz program, który umożliwi zamianę jednego ciągu znaków na inny ciąg znaków w pliku. Plik w którym będzie dokonywana zamiana, ma być podany jako parametr przy wywołaniu programu. Zamiana ma być wykonana bez względu na wielkość liter. Program ma składać się z następujących modułów: - obsługi klawiatury; - obsługi plików (wykrywanie, usuwanie, wyciąganie nazwy pliku); - zamiany ciągu znaków na inny w pliku. Poszczególne moduły programu powinny znajdować się w oddzielnych plikach. Łączenie poszczególnych plików ma nastąpić w module głównym programu.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwy poszczególnych modułów to **CW36Z.PAS**, **CW36Z1.PAS**, **CW36Z2.PAS** i **CW36Z3.PAS**.

Moduł obsługi klawiatury **CW36Z1.PAS**

```
{
Plik: cw36z1.pas
Program: Obsługa klawiatury
}

procedure KT_KursorTekst(X, Y, TS :Shortint; txtStr :String);
{ Wyświetla tekst wprowadzony z klawiatury }
var
     TT :Shortint;
  txtSpc :String;
begin
  { KT_KursorTekst }
  txtSpc:= '';
  for TT:= 0 to TS do
    txtSpc:= txtSpc+CHR(32);

  TextBackGround(BLACK);
  GotoXY(X, Y);
  Write(txtSpc);
  TextColor(WHITE);
  GotoXY(X, Y);
  Write(txtStr);
end;

function KN_KlawiszNacisnij :Integer;
{ Funkcja podaje kod (tj. numer) naciśniętego klawisza na klawiaturze }
var
  numKey :Integer;
     Ch :Char;
begin
  { KN_KlawiszNacisnij }
  repeat until KeyPressed;
    Ch:= ReadKey;
    numKey:= Ord(Ch);
```

```
    if (Ch = #0) then
    begin
      Ch:= ReadKey; { trap function keys: 359 - Help; 9 - TAB; 368 - F10 }
      numKey:= Ord(Ch)+300;
    end;
  KN_KlawiszNacisnij:= numKEY;
end;


function Klawisze(X, Y, TS :Shortint; txtStr :String) :String;
{ Funkcja obsługuje klawiaturę i kontroluje ilość wprowadzonych znaków z klawiatury }
var
  txtTekst :String;
    numLen :Shortint;
    numKey :Integer;
begin
  { Klawisze }
  TextBackGround(BLACK);
  TextColor(LightGRAY);
  GotoXY(X, Y);
  Write(txtStr+':');
  txtTekst:= '';
  repeat
    KT_KursorTekst(X+Length(txtStr)+1, Y, TS, txtTekst);
    numKey:= 0;
    numKey:= KN_KlawiszNacisnij;
    if (numKey = 8) then
    begin
      numLen:= 0;
      numLen:= Length(txtTekst);
      txtTekst:= Copy(txtTekst, 1, Length(txtTekst)-1);
    end
    else
    if (numKey = 27) then
    begin
      Klawisze:= '';
    end
    else
    if (numKey = 13) then
    begin
      Klawisze:= txtTekst;
    end
    else
    begin
      if (Length(txtTekst) < TS) then
      begin
        txtTekst:= txtTekst+CHR(numKey);
      end;
    end;
  until (numKey = 27) or (numKey = 13);
```

```
  TextBackGround(BLACK);
  TextColor(LightGRAY);
end;
```

Moduł obsługi plików (wykrywanie, usuwanie, wyciąganie nazwy pliku) **CW36Z2.PAS**

```
{
Plik: cw36z2.pas
Program: Pliki - Wykrywanie, Usuwanie
}

function FileExists(txtFileName :String) :Boolean;
{ Funkcja sprawdza, czy plik o podanej nazwie znajduje się fizycznie na dysku. Jeżeli tak, to
funkcja zwróci wartość TRUE (Prawda). W innym przypadku funkcja zwróci wartość FALSE
(Fałsz) }
var
  Info :SearchRec;
begin
  { FileExists }
  FindFirst(txtFileName, Directory, Info);
  FileExists:= FALSE;
  if (DosError = 0) then FileExists:= TRUE;
end;

function DeleteFile(txtFileName :String) :Boolean;
{ Funkcja usuwa plik o podanej nazwie. Gdy plik zostanie usunięty, funkcja zwraca wartość
TRUE (Prawda). W innym przypadku funkcja zwraca wartość FALSE (Fałsz) }
var
  TF :Text;
begin
  { DeleteFile }
  DeleteFile:= FALSE;
  if (FileExists(txtFileName) = TRUE) then
  begin
    Assign(TF, txtFileName);
    SetFAttr(TF, Archive); { For Windows: faArchive }
    Erase(TF);
    DeleteFile:= TRUE;
  end;
end;

function ExtractFileName(txtFileName :String) :String;
{ Funkcja wyciąga z nazwy pliku (nazwa pliku składa się z: - nazwy własnej,  - kropki, -
rozszerzenia) nazwę własną, np. „atari.txt” -> „atari”}
var
  numPos :Shortint;
begin
  ExtractFileName:= txtFileName;
  numPos:= 0;
```

```
  numPos:= Pos('.', txtFileName);
  if (numPos<>0) then ExtractFileName:= Copy(txtFileName, 1, numPos-1);
end;
```

Moduł zamiany ciągu znaków w pliku **CW36Z3.PAS**

```
{
Plik: cw36z3.pas
Program: Zamiana jednego ciagu znakow na inny ciag znakow w pliku
}

function IntToStr(X :Longint) :String;
{ Konwertuje liczbę na tekst }
var
  txtText :String;
begin
  { IntToStr }
  STR(X, txtText);
  IntToStr:= txtText;
end;

function LITERY_Male(txtStr :String) :String;
{ Zamienia duże litery na małe }

  function MalyZnak(chZnak :Char) :Char;
  var
    TT :Byte;
  begin
    for TT:= 65 to 90 do
      if (Chr(TT) = chZnak) then chZnak:= Chr(TT+32);
    MalyZnak:= chZnak;
  end;

var
  I :Byte;
begin
  for I:= 1 to Length(txtStr) do
    txtStr[I]:= MalyZnak(txtStr[I]);
  LITERY_Male:= txtStr;
end;

function jtbZnajdzZamien(txtText, txtFind, txtReplace :String) :String;
{ Zamienia jeden ciąg znaków na inny ciąg znaków }
var
  numPos, numLen: Integer;
begin
  { jtbZnajdzZamien }
  numLen:= 0;
  numLen:= Length(txtFind);
```

```
  while (Pos(LITERY_Male(txtFind), LITERY_Male(txtText)) > 0) do
  begin
    numPos:= 0;
    numPos:= Pos(LITERY_Male(txtFind), LITERY_Male(txtText));
    Delete(txtText, numPos, numLen);
    Insert(txtReplace, txtText, numPos);
  end;
  jtbZnajdzZamien:= txtText;
end;

function FILE_jbTextSaveToFile(txtFileName, txtText: String): Boolean;
{ Zapisuje dane do pliku tekstowego }
var
  TF :Text;
begin
  { TextSaveToFile }
  if (FileExists(txtFileName) = TRUE) then
  begin
    Assign(TF, txtFileName);
    Append(TF);
    Writeln(TF, txtText);
    Close(TF);
  end
  else
  begin
    Assign(TF, txtFileName);
    Rewrite(TF);
    Writeln(TF, txtText);
    Close(TF);
  end;
  FILE_jbTextSaveToFile:= TRUE;
end;

procedure OdczytajPlik(txtPlikZrodlo, txtPlikCel, txtF, txtR :String);
{ Odczytuje plik tekstowy, dokonuje zamiany ciągu znaków i po dokonanej zamianie zapisuje
go w nowym plik tekstowym o rozszerzeniu „OK” }
var
  txtCzytaj, txtZapiszText :String;
                    FT :Text;
                    TT :Longint;
begin
  { OdczytajPlik }
  if (FileExists(txtPlikZrodlo) = TRUE) then
  begin
    Assign(FT, txtPlikZrodlo);
    Reset(FT);
    TT:= 0;
    while not EOF(FT) do
    begin
```

```
    txtCzytaj:= '';
    ReadLn(FT, txtCzytaj);

    txtZapiszText:= '';
    txtZapiszText:= jtbZnajdzZamien(txtCzytaj, txtF, txtR);
    FILE_jbTextSaveToFile(txtPlikCel, txtZapiszText);
    GotoXY(4, 12);
    Write('Linia '+IntToStr(TT+1)+' zrobiona.');
    TT:= TT+1;
   end;
   Close(FT)
  end;
end;
```

Moduł główny programu **CW36Z.PAS**

```
uses Crt, Dos;

{$I cw36z1.pas}
{$I cw36z2.pas}
{$I cw36z3.pas}



var
  txtZnajdz, txtZamien, txtNazwaPliku :String;
                        okJestPlik :Boolean;
begin
  TextBackGround(BLACK);
  TextColor(LightGRAY);
  ClrScr;
  Writeln;
  Writeln('== Zamiana jednego ciagu znakow na inny ciag znakow w pliku ==');
  Writeln('Copyright(c)by Jan-Tadeusz Biernat');
  Writeln;

  if (ParamStr(1)<> '') then
  begin
    Writeln;
    Writeln('Nazwa pliku:'+CHR(32)+ParamStr(1));

    txtZnajdz:= '';
    txtZnajdz:= Klawisze(4, 8, 44, 'Szukany tekst');
    if (txtZnajdz<>'') then
    begin
      txtZamien:= '';
      txtZamien:= Klawisze(8, 10, 44, 'Zamien na');
      if (txtZamien<>'') then
      begin
```

```
      txtNazwaPliku:= '';
      txtNazwaPliku:= ExtractFileName(ParamStr(1))+'.ok';
      okJestPlik:= FALSE;
      okJestPlik:= FileExists(txtNazwaPliku);
      if (okJestPlik = TRUE) then DeleteFile(txtNazwaPliku);
      OdczytajPlik(ParamStr(1), txtNazwaPliku, txtZnajdz, txtZamien);
    end;
  end;
 end
 else
 begin
  Writeln('Jako parametr, nalezy podac nazwe pliku!');
 end;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

Ćwiczenie 5.7. Zamiana liczb na słowa – funkcja słownie

Napisz program, który umożliwi zamianę liczby na słowa. Program ma umożliwić wprowadzanie tylko cyfr, przecinka i znaku minus. Program ma składać się z następujących modułów: - obsługi klawiatury; - zamiana liczby na słowa. Poszczególne moduły programu powinny znajdować się w oddzielnych plikach. Łączenie poszczególnych plików ma nastąpić w module głównym programu.

Sposób wykonania
Program znajduje się w katalogu **TP7-CW**, nazwa poszczególnych modułów to **CW37Z.PAS**, **CW37Z1.PAS** i **CW37Z2.PAS**.

Moduł obsługi klawiatury **CW37Z1.PAS**

```
{
Plik: cw37z1.pas
Program: Obsługa klawiatury
}

procedure KT_KursorTekst(X, Y, TS :Shortint; txtStr :String);
{ Wyświetla tekst wprowadzony z klawiatury }
var
    TT :Shortint;
 txtSpc :String;
begin
 { KT_KursorTekst }
 txtSpc:= '';
 for TT:= 0 to TS do
  txtSpc:= txtSpc+CHR(32);

 TextBackGround(BLACK);
 GotoXY(X, Y);
```

```
  Write(txtSpc);
  TextColor(WHITE);
  GotoXY(X, Y);
  Write(txtStr);
end;

function KN_KlawiszNacisnij :Integer;
{ Funkcja podaje kod (tj. numer) naciśniętego klawisza na klawiaturze }
var
  numKey :Integer;
     Ch :Char;
begin
  { KN_KlawiszNacisnij }
  repeat until KeyPressed;
    Ch:= ReadKey;
    numKey:= Ord(Ch);
    if (Ch = #0) then
    begin
      Ch:= ReadKey; { trap function keys: 359 - Help; 9 - TAB; 368 - F10 }
      numKey:= Ord(Ch)+300;
    end;
  KN_KlawiszNacisnij:= numKEY;
end;


function Klawisze(X, Y, TS :Shortint; txtStr :String) :String;
{ Funkcja obsługuje klawiaturę i kontroluje ilość wprowadzonych cyfr oraz umożliwia
wprowadzenie tylko cyfr }
var
  txtTekst :String;
    numLen :Shortint;
    numKey :Integer;
begin
  { Klawisze }
  TextBackGround(BLACK);
  TextColor(LightGRAY);
  GotoXY(X, Y);
  Write(txtStr+':');
  txtTekst:= '';
  repeat
    KT_KursorTekst(X+Length(txtStr)+1, Y, TS, txtTekst);
    numKey:= 0;
    numKey:= KN_KlawiszNacisnij;
    if (numKey = 8) then
    begin
      numLen:= 0;
      numLen:= Length(txtTekst);
      txtTekst:= Copy(txtTekst, 1, Length(txtTekst)-1);
    end
    else
```

```
    if (numKey = 27) then
    begin
      Klawisze:= '';
    end
    else
    if (numKey = 13) then
    begin
      Klawisze:= txtTekst;
    end
    else
    begin
      if (Length(txtTekst) < TS) then
      begin
        if ((numKey > 47) and (numKey < 58)) or
           (numKey = 45) or (numKey = 44) then
        begin
          txtTekst:= txtTekst+CHR(numKey);
        end;
      end;
    end;
  until (numKey = 27) or (numKey = 13);
  TextBackGround(BLACK);
  TextColor(LightGRAY);
end;
```

Moduł zamiany liczb na słowa (tj. funkcja słownie) **CW37Z2.PAS**

```
{
 Plik: cw37z2

 == Slownie ==
 Copyright(c)by Jan-Tadeusz Biernat
}

function StrToInt(txtStr :String) :Longint;
{ Konwertuje tekst na liczby }
var
  numNumber, numErrCode :Integer;
begin
  { StrToInt }
  Val(txtStr, numNumber, numErrCode);
  StrToInt:= numNumber;
end;

function IntToStr(numNumer :Longint) :String;
{ Konwertuje liczbę na tekst }
var
   txtText: String;
begin
```

```
  txtText:= '';
  STR(numNumer, txtText);
  IntToStr:= txtText;
end;

function NUM_LikwidujSpacje(txtStr :String) :String;
{ Likwiduje spacje w wprowadzonym ciągu znaków }
var
      TT :Integer;
  txtText :String;
begin
  NUM_LikwidujSpacje:= '';
  if (txtStr<>'') then
  begin
    txtText:= '';
    for TT:= 1 to Length(txtStr) do
      if (txtStr[TT]<>CHR(32)) then
        txtText:= txtText+txtStr[TT];
    NUM_LikwidujSpacje:= txtText;
  end;
end;

function DodajZeroPrzed(txtZero :String) :String;
{ Dodaje cyfrę „0” przed pojedynczą cyfrą }
begin
  DodajZeroPrzed:= txtZero;
  if (Length(txtZero) = 1) then DodajZeroPrzed:= '0'+txtZero;
end;

function ZmienPrzecinekNaKropke(txtStr :String) :String;
{ Zamienia przecinek na kropkę }
var
  txtTekst :String;
    numPoz :Shortint;
begin
  ZmienPrzecinekNaKropke:= txtStr;
  if (txtStr<>'') then
  begin
    numPoz:= 0;
    numPoz:= Pos(',', txtStr);
    if (numPoz<>0) then
    begin
      txtTekst:= '';
      txtTekst:= Copy(txtStr, 1, numPoz-1)+'.'+
              Copy(txtStr, numPoz+1, Length(txtStr));
      ZmienPrzecinekNaKropke:= txtTekst;
    end;
  end;
end;
```

```
function NUM_PobierzTylkoLiczby(txtStr :String; chrPrzecinek :Char) :String;
{ Wybiera z ciągu znaków, tylko cyfry }
var
  txtCyfry :String;
       AA :Integer;
begin
  NUM_PobierzTylkoLiczby:= '0';
  if (txtStr<>'') then
  begin
    txtCyfry:= '';
    for AA:= 0 to Length(txtStr) do
     if (txtStr[AA] in ['0'..'9', chrPrzecinek]) then
       txtCyfry:= txtCyfry+txtStr[AA];
    NUM_PobierzTylkoLiczby:= txtCyfry;
    if (txtStr[1] ='-') then NUM_PobierzTylkoLiczby:= '-'+txtCyfry;
  end;
end;

function txtSlownieGramatyka(txtNumer :String; numPoz, numGrosz :Shortint) :String;
{ Zamienia trzy cyfry na tekst, np. 123 -> sto dwadzieścia trzy }
const
  MM: array[1..9, 1..4] of String[20] =
   (('sto ',         'dziesiec ',         'jeden ',    'jedenascie '),
    ('dwiescie ',    'dwadziescia ',      'dwa ',      'dwanascie '),
    ('trzysta ',     'trzydziesci ',      'trzy ',     'trzynascie '),
    ('czterysta ',   'czterdziesci ',     'cztery ',   'czternascie '),
    ('piecset ',     'piecdziesiat ',     'piec ',     'piętnascie '),
    ('szescset ',    'szescdziesiat ',    'szesc ',    'szesnascie '),
    ('siedemset ',   'siedemdziesiat ',   'siedem ',   'siedemnascie '),
    ('osiemset ',    'osiemdziesiat ',    'osiem ',    'osiemnascie '),
    ('dziewiecset ', 'dziewiecdziesiat ', 'dziewiec ', 'dziewietnascie '));

  NN: array[1..5, 1..3] of String[11] =
   (('zloty ',   'zlote ',    'zlotych '),
    ('tysiac ',  'tysiace ',  'tysiecy '),
    ('milion ',  'miliony ',  'milionow '),
    ('miliard ', 'miliardy ', 'miliardow '),
    ('bilion ',  'biliony ',  'bilionow '));

  PP: array [1..3] of String[7] = ('grosz', 'grosze', 'groszy');
var
  TT, JJ, numTPos: Shortint;
       txtSlow: String;
begin
  { txtSlownieGramatyka }
  txtSlownieGramatyka:= '';

  if (numPoz < 1) then numPoz:= 1;
  if (Length(txtNumer) = 1) then txtNumer:= '00'+txtNumer;
  if (Length(txtNumer) = 2) then txtNumer:= '0'+txtNumer;
```

```
  txtSlow:= '';
  if (Length(txtNumer) = 3) then
  begin
    if (StrToInt(Copy(txtNumer, 2, 2)) in [11..19]) and
      (StrToInt(Copy(txtNumer, 1, 1)) = 0) then
    begin
      txtSlownieGramatyka:= MM[StrToInt(Copy(txtNumer, 2, 2))-10, 4]+NN[numPoz, 3];
      if (numGrosz >0) then
        txtSlownieGramatyka:= MM[StrToInt(Copy(txtNumer, 2, 2))-10, 4]+PP[3];
    end
    else
    if (StrToInt(Copy(txtNumer, 2, 2)) in [11..19]) and
      (StrToInt(Copy(txtNumer, 1, 1)) > 0) then
    begin
      txtSlow:= '';
      txtSlow:= MM[StrToInt(Copy(txtNumer, 2, 2))-10, 4]+NN[numPoz, 3];
      if (numGrosz >0) then
        txtSlow:= MM[StrToInt(Copy(txtNumer, 2, 2))-10, 4]+PP[3];
      txtSlownieGramatyka:= MM[StrToInt(Copy(txtNumer, 1, 1)), 1]+txtSlow;
    end
    else
    begin
      txtSlow:= '';
      for TT:= 1 to Length(txtNumer) do
        for JJ:= 1 to 9 do
          if (Copy(txtNumer, TT, 1) = IntToStr(JJ)) then txtSlow:= txtSlow+MM[JJ, TT];
      numTPos:= 0;
      numTPos:= 1;
      if (StrToInt(Copy(txtNumer, 3, 1)) in [2..4]) then numTPos:= 2;
      if (StrToInt(Copy(txtNumer, 3, 1)) in [5..9]) or
        (StrToInt(Copy(txtNumer, 2, 1)) in [2..9]) and
        (StrToInt(Copy(txtNumer, 3, 1)) =1) or
        (StrToInt(Copy(txtNumer, 1, 1)) in [1..9]) and
        (StrToInt(Copy(txtNumer, 2, 1)) = 0) and
        (StrToInt(Copy(txtNumer, 3, 1)) = 1) or
        (StrToInt(Copy(txtNumer, 2, 1)) in [1..9]) and
        (StrToInt(Copy(txtNumer, 3, 1)) = 0) or
        (StrToInt(Copy(txtNumer, 1, 1)) in [1..9]) and
        (Copy(txtNumer, 2, 2) = '00') then
      begin
        numTPos:= 0;
        numTPos:= 3;
      end;
      txtSlownieGramatyka:= txtSlow+NN[numPoz, numTPos];
      if (numGrosz >0) then txtSlownieGramatyka:= txtSlow+PP[numTPos];
      if (Copy(txtNumer, 1, 3) = '000') then txtSlownieGramatyka:= '';
    end;
  end;
end;
```

```
function txtSlowniePisz(txtNumer, txtGrosz :String) :String;
var
  txtSlo, txtSGrosz :String;
            TT :Shortint;
begin
  { txtSlowniePisz }
  txtSlowniePisz:= '';
  txtSGrosz:= '';
  txtSGrosz:= 'i zero groszy';
  if (StrToInt(txtGrosz) >0) then
  begin
    txtSGrosz:= '';
    txtSGrosz:= 'i '+txtSlownieGramatyka(DodajZeroPrzed(txtGrosz), 1, 1);
  end;


  for TT:= 1 to 15-Length(txtNumer) do
    txtNumer:= '0'+txtNumer;

  txtSlo:= '';
  txtSlo:= txtSlownieGramatyka(Copy(txtNumer, 1, 3), 5, 0)+
        txtSlownieGramatyka(Copy(txtNumer, 4, 3), 4, 0)+
        txtSlownieGramatyka(Copy(txtNumer, 7, 3), 3, 0)+
        txtSlownieGramatyka(Copy(txtNumer, 10, 3), 2, 0)+
        txtSlownieGramatyka(Copy(txtNumer, 13, 3), 1, 0);

  if (txtSlo = '') then
  begin
   txtSlo:= '';
   txtSlo:= 'zero zlotych'+CHR(32);
  end;

  txtSlowniePisz:= txtSlo+txtSGrosz;
end;

function txtSlowniePLN(txtNumer :String) :String;
var
              numPoz: Shortint;
  txtSpacja, txtPrzecinek, txtCyfra,
  txtCyfraPrzed, txtCyfraPo: String;
begin
  { txtSlowniePLN }
  txtSpacja:= '';
  txtSpacja:= NUM_LikwidujSpacje(txtNumer);


  txtPrzecinek:= '';
  txtPrzecinek:= ZmienPrzecinekNaKropke(txtSpacja);
```

```
  txtCyfra:= '';
  txtCyfra:= NUM_PobierzTylkoLiczby(txtPrzecinek, '.');


  numPoz:= 0;
  numPoz:= Pos('.', txtCyfra);
  if (numPoz<>0) then
  begin
    txtCyfraPo:= '';
    txtCyfraPo:= DodajZeroPrzed(Copy(txtCyfra, numPoz+1, 2));

    txtCyfraPrzed:= '';
    txtCyfraPrzed:= Copy(txtCyfra, 1, numPoz-1);

    txtSlowniePLN:= txtSlowniePisz(txtCyfraPrzed, txtCyfraPo);
  end
  else
    txtSlowniePLN:= txtSlowniePisz(txtCyfra, '-1');
end;
```

Moduł główny programu **CW37Z.PAS**

```
uses Crt;

{$I cw37z1.pas}
{$I cw37z2.pas}



var
  txtLiczba :String;
begin
  TextBackGround(BLACK);
  TextColor(LightGRAY);
  ClrScr;
  Writeln;
  Writeln('== Zamiana liczb na slowa ==');
  Writeln('Copyright(c)by Jan-Tadeusz Biernat');
  Writeln;
  txtLiczba:= '';
  txtLiczba:= Klawisze(4, 5, 15, 'Podaj liczbe');
  if (txtLiczba<>'') then
  begin
    if (txtLiczba[1] = '-') then
    begin
      GotoXY(1, 7);
      Write('-'+txtSlowniePLN(Copy(txtLiczba, 2, Length(txtLiczba)-1)));
    end
```

```
    else
    begin
      GotoXY(1, 7);
      Write(txtSlowniePLN(txtLiczba));
    end;
  end
  else
  begin
    Writeln;
    Writeln;
    Write('Pa Pa!');
  end;
end.
```

Naciśnij kombinację klawiszy **CTRL+F9**, w celu uruchomienia programu.

6. Ćwiczenia do samodzielnego wykonania

Ćwiczenie 1
Omów różnice między stałą, a zmienną.

Ćwiczenie 2
Omów różnice między funkcją, a procedurą.

Ćwiczenie 3
Omów różnice między pętlą WHILE…DO, a REPEAT…UNTIL.

Ćwiczenie 4
Napisz program obliczający pole powierzchni kwadratu (wzór: a^2).

Ćwiczenie 5
Napisz program obliczający pierwiastki od liczby 1 do 100.

Ćwiczenie 6
Napisz program, który podniesie wylosowaną liczbę do potęgi trzeciej.

Ćwiczenie 7
Napisz wygaszać ekranu, który będzie umożliwiał wyświetlenie wprowadzonego napisu.

Ćwiczenie 8
Napisz wygaszać ekranu, który będzie umożliwiał wyświetlenie wylosowanego napisu z pośród wcześniej zdefiniowanych oraz będzie losował położenie tego napisu na ekranie.

Ćwiczenie 9
Napisz program, który poda ilość dni do końca bieżącego miesiąca.

Ćwiczenie 10
Napisz program wyświetlający na podstawie bieżącej daty imiona ludzi, którzy mają w danym dniu imieniny.

Ćwiczenie 11
Napisz program, który będzie umożliwiał dokonanie obliczeń przynajmniej z 4-ch podstawowych działań. Do wybrania działania należy wykorzystać menu. W czasie wprowadzania danych, należy zablokować wprowadzanie wszystkich znaków oprócz cyfr, znaku minus i przecinka.

Ćwiczenie 12
Do programu „Tabliczka mnożenia” dodaj licznik poprawnych i negatywnych odpowiedzi.

Ćwiczenie 13
Do programu „Tabliczka mnożenia” dodaj możliwość losowania pochwał i nagan za udzieloną odpowiedź.

Ćwiczenie 14
Dokonaj przeanalizowania programu „Przypominacz”.

Ćwiczenie 15

Napisz program, który podaje datę jutrzejszą i wczorajszą, na podstawie bieżącej daty.