

Algorytmika oraz programowanie w Języku Pascal

Algorytmika

Definicja 1.

Algorytm → jest to pewien ciąg czynności, który prowadzi do rozwiązania danego problemu.

Definicja 2.

Algorytm → to jednoznaczny przepis, opisujący krok po kroku sposób postępowania w celu rozwiązania pewnego problemu lub sposobu osiągnięcia jakiegoś celu.

Ilość kroków algorytmu zależy od tego, jak złożony jest problem, którego on dotyczy. Zawsze jednak liczba tych kroków będzie **liczbą skończoną**.

Algorytmy można przedstawiać m.in. następującymi sposobami:

- słowny opis
- schemat blokowy
- lista kroków
- drzewo algorytmu
- drzewo wyrażeń
- w pseudojęzyk
- w język programowania.

Cechy charakterystyczne poprawnego algorytmu:

1. **Poprawność** - dla każdego przypisanego zestawu danych, po wykonaniu skończonej liczby czynności, algorytm prowadzi do poprawnych wyników.

2. **Jednoznaczność** - w każdym przypadku zastosowania algorytmu dla tych samych danych otrzymamy ten sam wynik.

3. **Szczegółowość** - wykonawca algorytmu musi rozumieć opisane czynności i potrafić je wykonywać.

4. **Uniwersalność** - algorytm ma służyć rozwiązywaniu pewnej grupy zadań, a nie tylko jednego zadania.

Przykładowo algorytm na rozwiązywanie równań w postaci $ax + b = 0$ ma je rozwiązać dla dowolnych współczynników a i b , a nie tylko dla jednego konkretnego zadania, np. $2x + 6 = 0$

Etapy konstruowania algorytmu(programu):

1. Sformułowanie zadania.
2. Określenie danych wejściowych.
3. Określenie wyniku oraz sposobu jego prezentacji.
4. Ustalenie metody wykonania zadania.
5. Przy użyciu wybranej metody następuje zapisanie algorytmu.
6. Dokonujemy analizy poprawności rozwiązania.
7. Testowanie rozwiązania dla różnych danych.
8. Ocena skuteczności tegoż algorytmu.

Różne sposoby przedstawiania algorytmów

a) opis słowny

Jest na ogół pierwszym, mało ścisłym opisem sposobem rozwiązania problemu. Rozpoczyna się często dyskusją, w jaki sposób można rozwiązać postawione zadanie. Dyskusja służy do rozważań nad sposobem i technikami przydatnymi w rozwiązaniu problemu.

np. Opis słowny do algorytmu opisującego funkcję modułu (wartość bezwzględna).

Dla wartości dodatnich argumentu x funkcja przyjmuje wartość x , dla wartości ujemnych argumentu x funkcja przyjmuje wartość $-x$.

b) schemat blokowy

c) lista kroków

Poszczególne kroki zawierają opis operacji, które mają być wykonane przez algorytm. Mogą w nich również wystąpić polecenia związane ze zmianą kolejności wykonywanych kroków. Kolejność kroków jest wykonywana w kolejności ich opisu z wyjątkiem sytuacji gdy jedno z poleceń w kroku jest przejściem do kroku o podanym numerze. Budowa opisu algorytmu w postaci listy kroków jest następująca:

- ◆ tytuł algorytmu
- ◆ specyfikacja problemu
- ◆ lista kroków
- ◆ komentarze ujęte w nawiasy klamrowe {komentarz}

uwaga: Krok 0 może być opuszczony

*np. Lista kroków dla funkcji $SGN(x)$ czytaj *signum*.*

Algorytm obliczania wartości funkcji SGN(x)

Dane: Dowolna liczba rzeczywista x.

Wynik: Wartość funkcji

Krok 0. Wczytaj wartość danej x

Krok 1. Jeśli $x > 0$, to $f(x) = 1$. Zakończ algorytm.

Krok 2. { W tym przypadku $x \leq 0$. } Jeśli $x = 0$, to $f(x) = 0$. Zakończ algorytm.

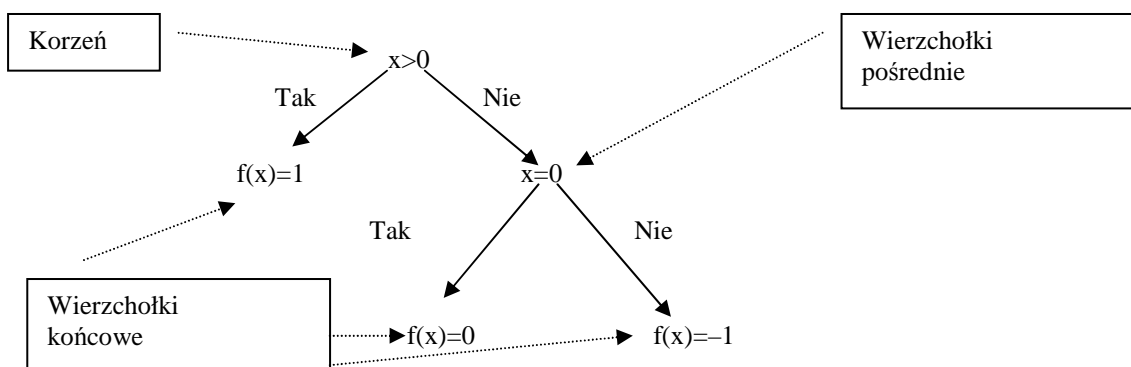
Krok 3. { W tym przypadku $x < 0$. } Mamy $f(x) = -1$. Zakończ algorytm.

d) drzewo algorytmu

Nazywany jest również drzewem obliczeń. Każde dwie drogi obliczeń mogą mieć tylko początkowe fragmenty wspólne, ale po rozejściu już nie spotykają.

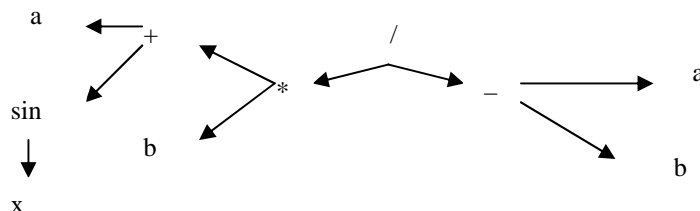
$$SGN(x) = \begin{cases} -1 & \text{dla } x < 0 \\ 0 & \text{dla } x = 0 \\ 1 & \text{dla } x > 0 \end{cases} \quad \text{się}$$

np. Drzewo algorytmu dla funkcji SGN(x).

**e) drzewo wyrażeń**

Stosowane do obliczeń wyrażeń arytmetycznych.

np. Wyrażenie $(a + \sin(x)) * b / (a - b)$

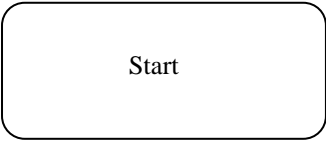
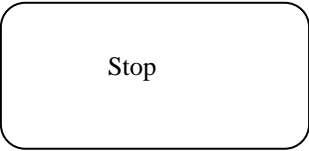

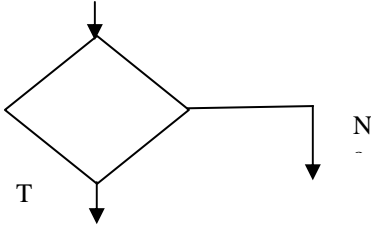
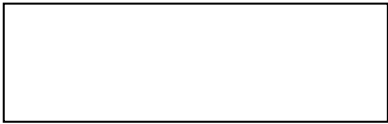
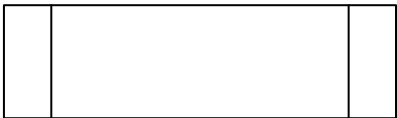


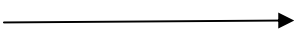
**f) program w języku programowania np. Pascal****g) pseudokod**

```
PROGRAM Wycieczka;
ZMIENNE punkty:naturalne;
        koszty, dofinansowanie:rzeczywiste;
ZACZNIJ;
    WPROWADŹ(PUNKTY,KOSZTY);
    JEŚLI punkty >=100 i punkty <= TO dofinansowanie :=1/3*koszty+0.2*koszty
        W PRZECIWNYM WYPADKU dofinansowanie:=0.2*koszty;
    WYPROWADŹ('Dofinansowanie wynosi:'dofinansowanie);
ZAKOŃCZ.
```

Specyfikacja problemu

Jest to dokładny opis problemu, który chcemy rozwiązać. Specyfikacja składa się z:

- ♦ **danych** oraz warunki jakie muszą spełniać,
- ♦ **wynik** oraz warunki jakie muszą spełniać (czyli związek pomiędzy danymi a wynikami).

Symbole stosowane w schematach blokowych.		
Początek algorytmu		W każdym algorytmie musi się znaleźć dokładnie jedna taka figura z napisem "Start" oznaczająca początek algorytmu. Blok symbolizujący początek algorytmu ma dokładnie jedną strzałkę wychodzącą.
Koniec algorytmu		W każdym algorytmie musi się znaleźć dokładnie jedna figura z napisem "Stop" oznaczająca koniec algorytmu. Najczęściej popełnianym błędem w schematach blokowych jest umieszczanie kilku stanów końcowych, zależnych od sposobu zakończenia programu. Blok symbolizujący koniec ma co najmniej jedną strzałkę wchodzącą.
blok wejścia wyjścia		Równoległobok jest stosowany do odczytu lub zapisu danych. W jego obrębie należy umieścić stosowną instrukcję np. Read(x) lub Write(x) (można też stosować opis słowny np. "Drukuj x na ekran"). Figura ta ma dokładnie jedną strzałkę wchodzącą i jedną wychodzącą. Jest to blok wejścia/wejścia wy/we I/O.
blok decyzyjny		Romb symbolizuje blok decyzyjny. Umieszcza się w nim jakiś warunek (np. " $x > 2$ "). Z dwóch wybranych wierzchołków rombu wyprowadzamy dwie możliwe drogi: gdy warunek jest spełniony (strzałkę wychodzącą z tego wierzchołka należy opatrzyć etykietą "Tak") oraz gdy warunek nie jest spełniony „Nie”. Każdy romb ma dokładnie jedną strzałkę wchodzącą oraz dokładnie dwie strzałki wychodzące.
blok decyzyjny		Jest to figura oznaczająca proces. W jej obrębie umieszczamy wszelkie obliczenia lub podstawienia . Proces ma dokładnie jedną strzałkę wchodzącą i dokładnie jedną strzałkę wychodzącą.
blok podprogramu		Ta figura symbolizuje proces, który został już kiedyś zdefiniowany. Można ją porównać do procedury, którą definiuje się raz w programie, by następnie móc ją wielokrotnie wywoływać. Warunkiem użycia jest więc wcześniejsze zdefiniowanie procesu. Podobnie jak w przypadku zwykłego procesu i tu mamy jedno wejście i jedno wyjście.
łącznik stronicowy		Koło symbolizuje tzw. łącznik stronicowy. Może się zdarzyć, że chcemy "przeskoczyć" z jednego miejsca na kartce na inne. Możemy w takim wypadku posłużyć się łącznikiem. Umieszczamy w jednym miejscu łącznik z określonym symbolem w środku (np. cyfrą, literą) i doprowadzamy do niego strzałkę. Następnie w innym miejscu kartki umieszczamy drugi łącznik z takim samym symbolem w środku i wyprowadzamy z niego strzałkę. Łączniki występują więc w parach, jeden ma tylko wejście a drugi wyjście.
łącznik międzystronicowy		Ten symbol to łącznik międzystronicowy. Działa analogicznie jak pierwszy, lecz nie w obrębie strony. Przydatne w złożonych algorytmach, które nie mieszczą się na jednej kartce.
element łączący		Poszczególne elementy schematu łączy się za pomocą strzałek. W większości przypadków blok ma jedną strzałkę wchodzącą i jedną wychodzącą.

Reguły rysowania schematów blokowych

- I. Po zbudowaniu schematu blokowego nie powinno być takich strzałek, które z nikąd nie wychodzą, lub do nikąd nie dochodzą.
- II. Każdy schemat blokowy musi mieć tylko jeden element startowy oraz co najmniej jeden element końca algorytmu.
- III. Element łączący(strzałki łączące) powinien być rysowany w poziomie i pionie, załamania pod kątem prostym.

Podział algorytmów.

Definicja algorytmu liniowego

Algorytmem liniowym nazywamy taki algorytm, który ma postać listy kroków wykonywanych zgodnie z ich kolejnością.

Algorytmy liniowe są zapisem obliczeń, które mają postać ciągu operacji rachunkowych wykonywanych bez sprawdzania jakichkolwiek warunków.

Algorytm z warunkami (rozgałęzieniami)

Ten typ algorytmu musi mieć bloki decyzyjne czyli bloki sprawdzania warunków.

Algorytmy numeryczne

Algorytmy, które wykonują działania matematyczne na danych liczbowych, nazywamy algorytmami numerycznymi.

Algorytm typu dziel i zwyciężaj

Dzielimy problem na kilka mniejszych, a te znowu dzielimy, aż ich rozwiązania staną się oczywiste,

Algorytmy iteracyjne

Iteracja jest to zapętlenie algorytmu, czyli wykonywania danych działań, dopóki warunek iteracji nie zostanie spełniony. Jest ona podstawą wszystkich choć troszkę bardziej złożonych algorytmów. Zazwyczaj ma ona składnię wykonuj "jakaś czynność" dopóki "jakieś wyrażenie logiczne".

Algorytmy rekurencyjne

Rekurencje wykorzystuje się do rozwiązywania problemów gdzie powtarza się czynność aby do niego dojść. Swoim działaniem przypomina iteracje. Jednak w tym przypadku **funkcja sama siebie wywołuje, dopóki nie otrzyma rozwiązania**, natomiast tam mieliśmy powtórzenie pewnej czynności określoną ilość razy.

Złożoność algorytmu- ilość zasobów potrzebnych do poprawnego działania danego algorytmu

Złożoności obliczeniowa-Algorytm wykonujący najmniejszą ilość operacji podstawowych w celu rozwiązania problemu.

Złożoność czasowa- Określa ilość operacji podstawowych potrzebnych do wykonania algorytmu o danej wielkości wejściowej.

Złożoność pamięciowa- Określa ilość przestrzeni pamięci wirtualnej potrzebnej do wykonania algorytmu z określonym zestawem danych wejściowych.

Uruchamianie PASCALA z dysku twardego

Przycisk Start opcja Uruchom w tym okienku wpisz:

c:\uczen\tp\turbo.exe

lub

opcja (lub skrót) Turbo Pascal

EDYTOR TP 6.0**Wgrzwanie okienka o żądanej nazwie pierwsza edycja(rozpoczęcie programu)****a)spółb pierwszy**

po wgraniu PASCALA, naciśnij klawisz F3 napisz swoją nazwę klawisz ENTER

b)spółb drugi

wciśnij klawisz F10 i wybierz opcję FILE klawiszami strzałek (← →) następnie rozwiń ją (ENTER) teraz wybierz opcję OPEN(strzałki dół i góra)edycyjnym i napisz swoją nazwę

c)spółb trzeci

wciśnij jednocześnie klawisze ALT+F a otrzymasz opcję FILE następnie rozwiń ją (ENTER) teraz wybierz opcję OPEN(strzałki dół i góra) i napisz swoją nazwę

d)spółb czwarty

najedź myszką na opcję FILE wciśnij lewy jej klawisz teraz najedź myszką na opcję OPEN i przyciśnij lewy klawisz myszy i w okienku edycyjnym i napisz swoją nazwę

e)spółb piąty

najedź myszką na opcję F3 OPEN w ostatniej linii ekranu wciśnij lewy przyciski napisz swoją nazwę

Nagrywanie napisanego tekstu (programu)

a)bez zmiany nazwy (tekst zostanie nagrany pod nazwą taką jaka jest obecnie)

- naciskamy klawisz F2
- wybieramy opcję FILE i dalej SAVE –najedź myszką na opcję F3 SAVE w ostatniej linii ekranu wciśnij lewy przycisk

b)ze zmianą nazwy (tekst zostanie nagrany pod nazwą taką jaką chcemy)

- wybieramy opcję FILE i dalej SAVE AS i napisz nazwę.

Wgrzwanie istniejącego tekstu z dysku**z klawiatury**

wciskamy klawisz F3 i pokazuje się okno i mamy możliwość

- piszemy nazwę zbioru który chcemy wgrać klawisz ENTER
- wciskamy klawisz Tab i strzałkami wybieramy nazwę zbioru do wgrania i wciskamy klawisz ENTER.

myszką

- kursor myszki na zbiór do wgrania i klawisz lewy myszy wciskamy szybko dwa razy. Jeśli niema interesującego zbioru w oknie to musisz przesunąć kursor na dolnej belce myszą, tak aby pokazał się interesujący zbiór.
- kursor myszki na zbiór do wgrania i lewy klawisz myszki nazwa zbioru pojawi się w okienku teraz wybieramy myszką opcję OPEN

Wgrzwanie pustego okna bez nazwy

- wybieramy opcję FILE i dalej NEW

Kompilowanie treści programu

- Jednoczesne wciśnięcie klawiszy Alt+F9
- Wybór opcji COMPILE z menu głównego a następnie podopcji COMPILE

uwaga:

program może być kompilowany :

- do pamięci
- na dysk wtedy tworzony jest program który możemy wykonać z poziomu systemu operacyjnego. Zbiór (program) będzie posiadać rozszerzenie EXE

Przełączanie sposobu kompilacji podopcja COMPILE z menu głównego Destination Memory → kompilacja do pamięci

Destination Disk → kompilacja na dysk

najechanie na opcję i ENTER powoduje przełączenie na przeciwne kompilowanie.

uwaga :

gdy kompilacja zakończyła się sukcesem (nie było błędów oknie pojawi się migający napis

Compile successfull:Press any key wciśnij ENTER i możesz pracować dalej

uwaga :

gdy jest błąd komputer podaje jego numer oraz opis po angielsku kursor ustawia się w pobliżu miejsca gdzie jest błąd. Wciśnij ENTER i popraw ten błąd. Jeśli nie wiesz dlaczego jest błąd do sprawdź w opisie błędów znajdującym się w tej instrukcji.

Uruchamianie programu

- jednocześnie wciśnięcie klawiszy Ctrl+F9
- Wybór opcji Run z menu głównego a następnie podopcji Run

Praca z oknami

a) zamknięcie okna edycyjnego (okno gdzie piszemy programy)

- Alt+F3
- kursor myszy na lewy górny narożnik i klawisz lewy myszy
- opcja delete ze spisu okien
- WINDOW a następnie CLOSE

b) zamknięcie okna nieedycyjnego (pozostałe okna)

- Alt+F3 lub Esc
- kursor myszy na lewy górny narożnik i klawisz lewy myszy
- wybór opcji CANCEL

c) uaktywnienie okna (wgranego do edytora)

- Alt + numer okna do edycji lub
- za pomocą spisu okien – podświetlamy okno w spisie i OK
- najeżdżamy myszką na powierzchnię okna które chcemy uaktywnić i lewy klawisz myszki

d) spis okien w edytorze

- Alt + 0
- WINDOW a następnie LIST

e) uaktywnienie następnego okna

- F6
- WINDOW a następnie NEXT

f) uaktywnienie poprzedniego okna

- Shift+F6
- WINDOW a następnie PREVIOUS

g) przemieszczanie okna

- Ctrl+F5 i następnie strzałki kursora (← →) na koniec ENTER
- myszką najeżdżamy kursorem na górną podwójną linię przyciskamy lewy przycisk i trzymając przemieszczamy po osiągnięciu pozycji puszczamy
- WINDOW a następnie SIZE/MOVE i następnie strzałki kursora (← →) na koniec ENTER

h) zmniejszanie okna

- Ctrl+F5 i potem Shift + strzałki kursora koniec ENTER
- myszką najeżdżamy na złączenie linii w dolnym prawym rogu okna wciskamy lewy przycisk myszy i przesuwamy mysz uzyskując powiększenie/pomniejszenie okna na koniec puszczamy
- WINDOW a następnie SIZE/MOVE i potem Shift + strzałki kursora koniec ENTER

i) powiększanie/pomniejszanie okna

- F5
- myszką najeżdżamy na strzałkę w górny prawym rogu okna i zatwierdzamy myszką

j) zmiana sposobu wyświetlania okien

- WINDOW potem CASADE – wyświetlanie okien jedno na drugim
- WINDOW potem TILE – wyświetlanie okien w równych rozmiarach

k) podglądanie wyniku programu

- WINDOW potem USER SCREEN – nasz do dyspozycji cały ekran
- ALT+F5
- WINDOW potem OUTPUT – otwarcie okna wyjściowego wraz z oknem edycyjnym

Użycie skrytki (CLIPBOARD)

Skrytka służy do przenoszenia części programu pomiędzy oknami.

W celu jej użycia wykonujemy:

- zaznaczamy blok (czyli tę część programu, którą chcemy przenieść)
- wykonujemy EDIT potem COPY (lub Ctrl+Ins) jest to wgranie bloku do skrytki
- uaktywniamy okno gdzie chcemy przenieść blok
- wykonujemy EDIT potem PASTE (lub Shift+Ins) jest to przeniesienie bloku do nowego okna

– **masz jeszcze możliwości** –

- oglądnięcie zawartości skrytki EDIT potem SHOW CLIPBOARD
- skasowanie skrytki EDIT potem CLEAR (lub CTRL+DEL)
- przeniesie do skrytki ze skasowaniem bloku EDIT potem CUT (lub SHIFT+DEL)
- kopiowanie przykładów z HELP do skrytki najpierw EDIT potem COPY EXAMPLE

Chwilowe zakończenie pracy w PASCALU – przejście do DOSu

- najpierw opcja z menu głównego FILE potem DOS shell
- gdy chcesz powrotu do PASCALA napisz EXIT i klawisz ENTER

Zakończenie pracy w PASCALU

- Alt+X

- najpierw opcja z menu głównego FILE potem Exit

Przerwanie działania programu

- Ctrl+Break (Pause)

Śledzenie wykonywania programu

–podgląd wartości zmiennych, wyrażeń

Opcja Debug podopcja Watches

- Add watch (CTRL+F7) → dodawanie wyrażeń do podglądu
- Delete watch → kasowanie wyrażeń do podglądu
- Edit watch → poprawianie wyrażeń do podglądu
- Remove all watches → kasowanie wszystkich wyrażeń

–ustalanie punktów kontrolnych

a) Opcja Debug podopcja Toggle breakpoint (CTRL+F8)→powoduje wstawienie do treści programu punktu w którym program zostanie przerwany (linia w programie zostanie podświetlona)

b) Opcja Debug podopcja Breakpoints→powoduje wyświetlenie okna do dodatkowych czynności z punktami przerwania

*Edit → edycja punktów przerwań

- **Condition → podawanie warunku na przerwanie
- **Pass count → ile razy ma nastąpić pominięcie przerwania
- **File name → zbiór do przerwania
- **Line number → numer li ni do przerwania

*Delete → kasowanie punktów przerwań

*View → sprawdzanie punktu przerwania

*Clear all → kasowanie wszystkich punktów przerwania

–zmazywanie punktów kontrolnych

*jeden punkt kontrolny → opcja Debug podopcja Breakpoint i teraz Delete

*wszystkie punkty kontrolne → CTRL+F2

–wykonanie krokowe programu

*F7–wykonanie krokowe bez przejścia do procedury

*F8–wykonanie krokowe programu

Włączanie kalkulatora oraz wyświetlanie aktualnych wartości zmiennych

Ctrl+F4 lub opcja Debug następnie Evaluate/modify Pojawi się okno dialogowe

Expression –podajemy wyrażenie do obliczenia lub nazwę zmiennej, której chcemy poznać wartość

Result–w tym miejscu pokaże się obliczona wartość wyrażenia lub wartość zmiennej

Polecenia edytora

uwaga: znak ^ oznacza klawisz CTRL

^KD– (taki zapis oznacza, że najpierw trzymamy klawisz CTRL i raz dociskamy K następnie puszczaemy dwa te klawisze i przyciskamy klawisz D)

DEL – usunięcie znaku pod kursorem

BACKSPACE – usunięcie znaku na lewo od kursora

poruszanie się po tekście

^E – kursor wiersz do góry (jednocześnie klawisze CTRL i E)

^X – kursor wiersz do dół

^S – kursor w lewo

^D – kursor w prawo

cztery poprzednie polecenia można również osiągnąć strzałkami kursora

^A – kursor słowo w lewo lub Ctrl+←

^F – kursor słowo w prawo lub Ctrl+→

^C – kursor ekran na dół lub PgDn

^R – kursor ekran do góry lub PgUp

^W – tekst o jeden wiersz do góry

^Z – tekst o jeden wiersz do dołu

^G – usunięcie znaku wyróżnionego przez kursor

^T – usunięcie znaków od znaku wyróżnionego przez kursor do końca słowa

^QE – pierwszy wiersz ekranu lub ^+Home

^QX – ostatni wiersz ekranu lub ^+End

^QS – pierwszy znak wiersza lub Home

^QD – ostatni znak wiersza lub End

^QC – koniec tekstu lub ^+PgDn

^QR – początek tekstu lub ^+PgUp

operacje na całych wierszach

^Y – usunięcie wiersza z kursorem

^N – pusty wiersz

wyszukiwanie i zastępowanie

^QF – wyszukiwanie tekstu lub opcja Search i Find pojawi się okno

Text to find–wpisujemy tekst do szukania

Options

[] Case sensitive

rozróżnianie małych i dużych liter

[] Whole words only

całe słowa

[X] Regular expression

tylko wyrażenia regularne

Scope

(.) Global

przeszukiwanie w całym tekście

() Selected text

zaznaczony fragment tekstu

Direction

(.) Forward

szukanie do przodu

() Backward

szukanie w tył

Origin

() From cursor

przeszukiwanie od kursora

(.) Entire scope

przeszukiwanie jak zaznaczonego Scope

uwaga1:opcja aktywna w oknie Options zaznaczona jest przez X

uwaga2:opcja aktywna w oknie Direction,Scope,Origin zaznaczona jest przez (.) czyli kropka w nawiasie okrągłym

uwaga3: zaznaczenie aktywnego działania opcji przez podanie pierwszej litery prócz Forward gdzie należy wcisnąć klawisz d, zmiana opcji do wyboru klawisz Tab. Aktywna opcja zaznaczona przez >>

^QA – wyszukiwanie i zastępowanie tekstu ENTER lub Search i Replace

Pojawi się okno

Text to find–wpisujemy tekst do szukania

New text – nowy tekst

Options

[] Case sensitive

rozróżnianie małych i dużych liter

[] Whole words only

całe słowa

[X] Regular expression

tylko wyrażenia regularne

[] Prompt on replace

pytanie przy każdej wymianie

Scope

(.) Global

przeszukiwanie w całym tekście

() Selected text

zaznaczony fragmentu tekstu

Direction

(.) Forward

szukanie do przodu

() Backward

szukanie w tył

Origin

() From cursor

przeszukiwanie od kursora

(.) Entire scope

przeszukiwanie jak zaznaczonego

Scope

operacje na blokach

trzy sposoby **zaznaczenia** bloku(blok jest wyróżnioną częścią programu przez podświetlenie)

a)ustawiamy się kursorem w miejscu ,które ma być początkiem bloku i trzymając klawisz SHIFT strzałkami zaznaczamy blok

b)najpierw jedziemy na miejsce ,które ma być początkiem bloku i wciskamy ^KB(brak reakcji komputera) potem jedziemy na miejsce które ma być końcem bloku i ^KK zapali się blok

c)ustawiamy się w miejscu ,które ma być początkiem bloku i myszką trzymając jej lewy przycisk przemieszczamy się po ekranie zaznaczając blok

^KB – zaznaczenie początku bloku

^KK – zaznaczenie końca bloku

^KC – kopiowanie bloku od miejsca gdzie jest kursor lub ^+Ins

^KV – przeniesienie bloku do miejsca gdzie jest kursor lub Shif+Ins

^KY – kasowanie bloku

^KT – zaznaczenie pojedynczego słowa jako blok

^QB – na początek bloku

^QK – na koniec bloku

^KH – **gaszenie/zapalenie** zaznaczenia bloku

lub

myszą: najeżdżamy kursorem myszy na miejsce poza blokiem i wciskamy lewy przycisk

^KR – wgranie bloku z dysku

READ BLOK FROMFILE:nazwa bloku ENTER

^KW – nagranie bloku na dysk

WRITE BLOK FROM FILE: nazwa zbioru ENTER

^KP – drukowanie tekstu programu lub opcja File a następnie Print

^KU – przeniesienie bloku o jedną kolumnę w lewo

^KI – przeniesienie bloku o jedną kolumnę w prawo

polecenia różne

^V – przełącznik zastępowania tekstu na wstawianie lub Ins

(tryb NAP na DOP)

^K-n – zapamiętanie położenia kursora n–liczba naturalna

^Q-n – przeniesienie kursora do zapamiętanego położenia kursora

^I – przeniesienie kursora do najbliższej tabulacji

^L – powtórzenie ostatniej dyrektywy ^QF

^U – zaniechanie dowolnej dyrektywy

^Q0 – włączenie i wyłączenie automatycznego wcinania tekstu

^QP – poprzednia pozycja kursora

^QL – ignorowanie zmian wprowadzonych w bieżącym wierszu

^P – wprowadzenie znaku sterującego do tekstu

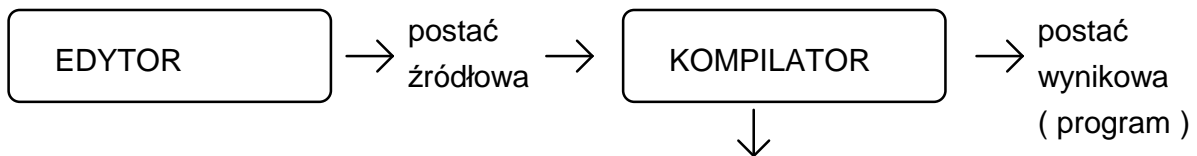
PIERWSZA CZĘŚĆ WYKŁADU Z PASCALA

Organizacja systemu PASCALA

System TURBO PASCAL składa się z:

1. **EDYTORA**– służy do tworzenia tekstu programu w języku PASCAL, traktowanej jako ciąg znaków (litera+liczby+znaki specjalne). Tekst tego programu nazywany jest programem w postaci źródłowej.

2. **KOMPILATORA**–program tłumaczący program w postaci źródłowej na program w postaci wynikowej (jest to ciąg operacji maszynowych zakodowanych w postaci liczb dwójkowych. Postać wynikowa jest gotowa do wykonania. Gdy program źródłowy ma błędy kompilator sygnalizuje błąd poprzez wskazanie jego miejsca oraz podanie jego numeru.



Rodzaje kompilacji

- ♦ kompilacja do pamięci → program może być uruchamiany z poziomu Pascala
- ♦ kompilacja na EXE → otrzymasz program, który może być wykonywany bez Pascal z DOSu lub okienka DOSu w Windowsa

Interpreter

Istnieją języki programowania, które nie wykonują kompilacji całego programu przed jego wykonaniem ale czytają po kolei wszystkie linijki treści programu i każdą linijkę osobno wykonują czyli tłumaczą na kod maszynowy jedną linię a potem następną i tak aż do wykonania całego programu. Tak działają języki programowania nazywane interpreterami.

Konsolidator

Konsolidacja programu nazywana inaczej linkowaniem polega na tym, że gdy oprócz programu masz jeszcze inne pliki np. z przydatnymi funkcjami i należy połączyć Twój program w jeden wykonywalny program to jest właśnie konsolidacja.

STRUKTURA PASCALA

- 1) Nagłówek programu.
- 2) Opcje kompilatora.
- 3) Dyrektywy włączeń programowych.
- 4) Deklaracje etykiet.
- 5) Deklaracje stałych.
- 6) Deklaracja typów.
- 7) Deklaracja zmiennych.
- 8) Procedura (podprogram).
- 9) Deklaracja funkcji.
- 10) Część operacyjna.

OPIS STRUKTURY PASCALA

ad 1) → NAGŁÓWEK

Nagłówek składa się z PROGRAM nazwy programu.

Uwaga!

Każda linia (instrukcja) w Pascalu kończy się średnikiem (;)

(są wyjątki BEGIN VAR CONST USES....)

np.

PROGRAM OBLICZ;

ad 5) → DEKLARACJA STAŁYCH

Rozpoczynamy deklarację stałych od słowa CONST

np.

PROGRAM NIC;

CONST
PI=3.14159;

ad 7) → DEKLARACJA ZMIENNYCH

Rozpoczynamy deklarację zmiennych od słowa VAR.

Podstawowe rodzaje zmiennych:

- typu **BOOLEAN**

przyjmują tylko dwie wartości:

false – fałsz

true – prawda

- typu **INTEGER**– liczby całkowite z przedziału $<-32768, 32767>$ stała zależna od komputera.

- typu **REAL** – liczby rzeczywiste

UWAGA1!

Do zapisu stosuje się kropkę zamiast przecinka np. 45.8

UWAGA2!

Stosuje się także zapis w postaci wykładniczej

np. $3.14E2 = 3.14 \cdot 10^2 = 314$ ^ – oznacza do potęgi

$4500E-3 = 4500 \cdot 10^{-3} = 4.5$

- typu **CHAR**

zmienna znakowa–mogą nimi być litery A–Z liczby traktowane jako znaki 1–9 oraz inne znaki np. !@#\$%^&*() itp.

- typu **STRING[n]**–ciągi znaków n–długość ciągu znakowego $n \leq 255$

np.

PROGRAM TYLK;

CONST

A=30;

VAR

p,q:BOOLEAN;

K:INTEGER;

c,ZMIENNA:REAL;

ZNAK:CHAR

napis:string[100]

ad 10)→CZĘŚĆ GŁÓWNA PROGRAMU (OPERACYJNA)

Program główny zaczyna się od BEGIN kończy na END.(kropka końcowa konieczna).

np.

PROGRAM PRZYKŁAD;

CONST

ALFA=83.3;

VAR

WIEK:INTEGER;

BEGIN

WRITELN('PRZYKŁAD');

END.

Opis instrukcji WRITE,WRITELN

WRITE (zmienne lub teksty).

znaczenie:

Wyprowadzenie na urządzenie wyjściowe (monitor, drukarkę) zmiennych lub tekstów bez przejścia kursora do nowej linii. Cursor jest to miejsce na ekranie monitora gdzie będziemy wpisywać znaki (miejsce to jest podświetlone lub jest to mrugający prostokąt)

–wyprowadzenie tekstu ujętego w apostrofy

np. WRITE ('To ja Pascal');

–wyprowadzenie zawartości zmiennych

np. WRITE ('W komórce a=',a);

∇

∇

wydruk tekstu wydruk zawartości komórki a (zmiennej).

–wykonanie obliczeń z wykorzystaniem zawartości komórek

np. WRITE (' S = ', V*T);

\vee \vee
 tekst iloczyn zawartości komórek (zmiennych).

WRITELN(zmienne lub teksty).

znaczenie:

Wyprowadzenie na monitor zmiennych lub tekstów z przejściem kursora do nowej linii(kolejny wydruk będzie zaczynał się od nowej linii na ekranie monitora)

WRITELN;

znaczenie:

instrukcja ta wyprowadza pustą linię na ekran monitora.

Kody ASCII

ASCII [aski] (ang. American Standard Code for Information Interchange) - 7-bitowy kod przyporządkowujący liczby z zakresu 0-127 literom (alfabetu angielskiego), cyfrom, znakom przestankowym i innym symbolom oraz poleceniom sterującym. Przykładowo litera "a" jest kodowana liczbą 97, a znak spacji - 32.

Litery, cyfry oraz inne znaki drukowane tworzą zbiór znaków ASCII. Jest to 95 znaków o kodach 32-126. Pozostałe 33 kody (0-31 i 127) to tzw. kody sterujące służące do sterowania urządzeniem odbierającym komunikat, np. drukarką czy terminalem.

Specjalne znaki trybu tekstowego

Są to znaki, których nie ma na klawiaturze np. Ę. lecz istnieją jako kod ASCII. Aby wywołać te znaki potrzeba jest kombinacji klawisza ALT.

ALT+xxx

gdzie xxx to są cyfry z klawiatury numerycznej. Trzymamy ALT i wystukujemy cyfry z klawiatury numerycznej → gdy puścimy ALT zobaczymy znak specjalny.

128	Ç	144	É	161	í	177	␣	193	⌞	209	⌘	225	β	241	±
129	ü	145	æ	162	ó	178	␣	194	⌞	210	⌘	226	Γ	242	≥
130	é	146	Æ	163	ú	179		195	⌞	211	⌘	227	π	243	≤
131	â	147	ô	164	ñ	180	⌞	196	—	212	⌞	228	Σ	244	∫
132	ä	148	ö	165	Ñ	181	⌞	197	+	213	⌞	229	σ	245	∫
133	à	149	ò	166	²	182	⌞	198	⌞	214	⌞	230	μ	246	÷
134	â	150	û	167	°	183	⌞	199	⌞	215	⌞	231	τ	247	≈
135	ç	151	ù	168	¿	184	⌞	200	⌞	216	⌞	232	Φ	248	°
136	ê	152	—	169	—	185	⌞	201	⌞	217	⌞	233	⊖	249	.
137	ë	153	Ö	170	⌞	186	⌞	202	⌞	218	⌞	234	Ω	250	.
138	è	154	Û	171	½	187	⌞	203	⌞	219	■	235	δ	251	√
139	ï	156	£	172	¼	188	⌞	204	⌞	220	■	236	∞	252	—
140	î	157	¥	173	ı	189	⌞	205	=	221	■	237	φ	253	²
141	ì	158	—	174	«	190	⌞	206	⌞	222	■	238	ε	254	■
142	Ä	159	ƒ	175	»	191	⌞	207	⌞	223	■	239	∩	255	
143	Å	160	á	176	␣	192	⌞	208	⌞	224	α	240	≡		

Source: www.LookupTables.com

PRZYKŁAD 1) (wpisać do komputera i sprawdzić działanie programu)

Wpisz do zeszytu znaczenie instrukcji CLRSCR;

Wypisz na ekranie monitora nazwę szkoły (możesz napisać nazwę swojej szkoły). Wokół nazwy szkoły jest obramowanie w postaci gwiazdek.

Nagraj program na pendrive (dyskietkę) ucznia pod odpowiednią nazwą: cztery pierwsze litery nazwiska (nie stosuj polskich liter) podkreślenie numer przykładu np. kowa_p1

```

PROGRAM ADRES;
USES
  CRT;
  { DOŁĄCZENIE MODUŁÓW }
  { MODUŁ EKRANU }
BEGIN
  CLRSCR;

```

```

WRITELN;
WRITELN('*****');
WRITELN('*ZSZ ENERGETYCZNYCH      *');
WRITELN('*REJA 25                      *');
WRITELN('*****');
REPEAT UNTIL KEYPRESSED; {ZATRZYMANIE PROGRAMU}
END.

```

Instrukcja **CLRSCR** → kasowanie ekranu w trybie tekstowym.

Uwagi dotyczące pisania programów

UWAGA1:

Przy pisaniu programów w języku PASCAL nie jest istotne czy program jest opisany **dużymi** czy **małymi** literami.

UWAGA2:

W treści programu robione są **WCIECIA** w celu zwiększenia czytelności programu. Gdy będziesz pisał programy rób zawsze wcięcia.

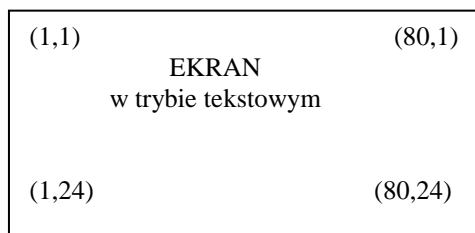
UWAGA3:

W treści programu na końcu instrukcji piszemy średnik (;)

ZADANIE 1.

Zmodyfikuj poprzedni program tak aby wizytówka wyświetliła się na **środku ekranu**.

Użyć instrukcji gotoxy(p,q), np. gotoxy(10,10);write('*'); narysuje gwiazdkę w miejscu współrzędnych ekranu (10,10). Przerysuj do zeszytu schemat ekranu, patrz poniżej oraz opisz instrukcję gotoxy(p,q).



Instrukcja **gotoxy(p,q)** → przeniesienie kursora ekranu w trybie tekstowym do miejsca ekranu o współrzędnych (p,q). Nagraj program na pendrive(dyskietkę) ucznia pod odpowiednią nazwą: cztery pierwsze litery nazwiska (nie stosuj polskich liter) podkreślenie numer zadania np. kowa_z1. Dokonaj kompilacji na EXE i uruchom ten program z poziomu DOS.

ZADANIE 2.

Zmodyfikuj poprzedni program tak aby wizytówka wyświetliła się w obramowaniu z ramki podwójnej wykonanej ze znaków specjalnych (kodów ASCII). Kolor tła jasnoszary. Tekst w dwóch liniach pisany dwoma różnymi kolorami(nie kolorem tła).

Zapisz w zeszycie co to są kody ASCII.

Zapisz w zeszycie znaczenie instrukcji:

- TEXTMODE();
- WINDOW();
- TEXTBACKGROUND();
- TEXTCOLOR();
- SOUND();
- DELAY();
- NOSOUND;
- REPEAT UNTIL KEYPRESSED;

na podstawie przykładu programu DEMONSTRACJA MODUŁU CRT

Nagraj program na pendrive(dyskietkę) ucznia pod odpowiednią nazwą: cztery pierwsze litery nazwiska (nie stosuj polskich liter) podkreślenie numer zadania np. kowa_z2. Dokonaj kompilacji na EXE i uruchom ten program z poziomu DOS.

ZADANIE 3. (praca domowa)

Napisać program rysujący domek ze znaków dostępnych z klawiatury oraz zastosuj specjalne znaki trybu tekstowego. Użyj kolorów, instrukcji gotxy(p,q).

Nagraj program na pendrive(dyskietkę) ucznia pod odpowiednią nazwą: cztery pierwsze litery nazwiska (nie stosuj polskich liter) podkreślenie numer zadania np. kowa_z3. Dokonaj kompilacji na EXE i uruchom ten program z poziomu DOS.

SPRAWDZIAN (przykładowy zestaw czas 15 minut na program 75 minut na edytora PASCALA)

- napisać program rysujący czołg ze znaków dostępnych z klawiatury
- znajomość systemu PASCALA (uruchamianie PASCAL, zakończenie pracy w PASALU, praca z oknami, operacje na blokach, wyszukiwanie i zastępowanie, poruszanie się po tekście, zapisywanie programu na dysku, wczytywanie programu do edytora, nowy tekst programu, kompilowanie na EXE).

DEMONSTRACJA MODUŁU CRT

PROGRAM DEMONSTRACJA_MODULU_CRT;

```

USES CRT;           { DOŁĄCZENIE PAKIETU CRT –MODUŁU EKRANU }
VAR
  I:INTEGER;
  Q:CHAR;
BEGIN
  CLRSCR;           { CZYSZCZENIE EKRANU }
  TEXTMODE(3);      { WYBÓR RODZAJU TRYBU TEKSTOWEGO }
                    { 0– BW40 ,TRYB CZARNO–BIAŁY 40*25 }
                    { 1– CO40 ,TRYB KOLOROWY 40*25 }
                    { 2– BW80 ,TRYB CZARNO–BIAŁY 80*25 }
                    { 3– CO80 ,TRYB KOLOROWY 80*25 }
                    { 7– MONO ,TRYB STEROWNIKA MONO 80*25 }
                    { 256– FONT8*8 ,TRYB 43– LUB 50–WIERSZOWY }
  TEXTBACKGROUND(0); { ZMIANA KOLORU TŁA }
  TEXTCOLOR(15);     { ZMIANA KOLORU TEKSTU }
  { BLACK =0 CZARNY } { BLUE =1 NIEBIESKI } { GREEN =2 ZIELONY }
  { CYAN =3 TURKUSOWY } { RED =4 CZERWONY } { MAGENTA =5 KARMAZYNOWY }
  { BROWN =6 BRĄZOWY } { LIGHTGRAY =7 JASNOSZARY }
  { DARKGRAY =8 CIEMNOSZARY } { LIGHTBLUE =9 JASNONIEBIESKI }
  { LIGHTGREEN =10 JASNOZIELONY } { LIGHTCYAN =11 JASNOTURKUSOWY }
  { LIGHTRED =12 JASNO CZERWONY } { LIGHTMAGENTA=13 JASNOKARMAZYNOWY }
  { YELLOW =14 ŻÓŁTY } { WHITE =15 BIAŁY } { BLINK =128 BIT MIGANIA }
  CLRSCR;
  WRITE('TO JEST DEMO CRT –PAKIET ');
  DELAY(1000); {ZATRZYMANIE KOMPUTERA NA 1000= JEDNA SEKUNDA }
  WINDOW(5,5,40,20); { WYBORU OKNA O PODANYCH WSP. NAROŻY }
                    { (LEWE GÓRNE , PRAWE DOLNE ) }
  TEXTBACKGROUND(CYAN);
  TEXTCOLOR(WHITE);
  CLRSCR; { UWAGA ABY UJAWNIEĆ KOLOR TŁA ORAZ PISMA ONA NALEŻY }
          { SKASOWAĆ EKRAN }
  GOTOXY(1,1); { PRZESUNIĘCIE KURSORA DO ZNAKU O WSP.(X,Y) }
               { W OKNIE TEKSTOWYM }
  WRITELN('TUTAJ BĘDZIE NAPIS W OKNIE ');
  GOTOXY(3,3);
  WRITELN(' INFORMATYKA – TO JEST TO!');
  FOR I:=1 TO 10 DO
  BEGIN
    SOUND(120+I*10); { WŁĄCZENIE EMISJI SYGNAŁU DŹWIĘKOWEGO }
                    { O ZADANEJ CZĘSTOTLIWOŚCI }
    DELAY(500);
    NOSOUND; { WYŁĄCZENIE EMISJI SYGNAŁU DŹWIĘKOWEGO }
  END;
  GOTOXY(5,12);
  TEXTCOLOR(BLINK); { WŁĄCZENIE MIGANIA }
  WRITELN('JEŚLI DALEJ WCIŚNIJ DOWOLNY KŁAWISZ');
  Q:=READKEY; { READKEY – PROGRAM CZYTANIA ZNAKU Z KŁAWIATURY }

```


CZĘŚĆ DRUGA WYKŁADU Z PROGRAMOWANIA W JĘZYKU PASCAL OPERATORY I WYRAŻENIA ALGEBRAICZNE

KOMENTARZE w treści programu

W celu lepszej czytelności tekstu programu używane są komentarze.

Tekst ujęty jest w nawiasy klamrowe

- { komentarz } lub
- (*komentarz*)

zapisy są równoważne.

PRZYKŁAD (nie wpisywać do komputera)

```
PROGRAM KOMEN; {POCZĄTEK PROGRAMU }
```

```
(*TERAZ POCZĄTEK TREŚCI OPERACYJNEJ*)
```

```
BEGIN
```

```
  WRITELN('TO JEST PRZYKŁAD NA ZASTOSOWANIE KOMENTARZY');
```

```
  { TERAZ KONIEC PROGRAMU }
```

```
END.
```

OPERATORY

+	dodawanie	<	mniejsze
-	odejmowanie	>	większe
*	mnożenie	<=	mniejsze lub równe
/	dzielenie	>=	większe lub równe
<>	nierówne		
DIV			dzielenie całkowite
MOD			branie reszty z dzielenia
ABS(x)			wartość bezwzględna
SQR(x)			kwadrat
SQRT(x)			pierwiastek kwadratowy
LN(x)			logarytm naturalny
EXP(x)			funkcja wykładnicza o podstawie e
SIN(x)			x podajemy w radianach
COS(x)			
ARCTAN(x)			arcustangens–funkcja, która na podstawie wartości tangensa podaje wartość kąta lecz w radianach
SUCC(x)			następnik
PRED(x)			poprzednik
ROUND(x)			zaokrąglenie do najbliższej liczby całkowitej
TRUNC(x)			zaokrągla w kierunku zera (dla liczb dodatnich oznacza to "w dół", dla liczb ujemnych - "w górę") podaną liczbę rzeczywistą i zwraca ją w postaci liczby całkowitej. Innymi słowy funkcja Trunc obcina część ułamkową.
ODD(x)			funkcja nieparzystości, true dla nieparzystych, false dla parzystych
INT(x)			zwraca część całkowitą liczby rzeczywistej x
FRAC(x)			zwraca część ułamkową liczby rzeczywistej x
NOT			negacja
AND			iloczyn logiczny
OR			suma logiczna

PRZYKŁADY

7 DIV 4 = 1	ARCTAN(1)=PI/4=0.785=(45 STOPNI)
7 MOD 4 = 3	
ABS(-5) = 5	SIN(1.57)=1 (1.57=PI/2=90 STOPNI)
SQR(2) = 4	
SUCC(7) = 8	INT(3.56)=3.00
PRED(8) = 7	INT(-4.33)= -4.00
ROUND(2.51) = 3	FRAC(2.45)=0.45
ROUND(2.49) = 2	
TRUNC(2.7) = 2	
TRUNC(-2.7) = -2	

Priorytety działań

priorytety (od najwyższego) przedstawiają się następująco:

– (jednoargumentowy zmiana znaku)

* / div mod

+ - (dwuargumentowe)

Ogólnie, operatory o tych samych priorytetach wykonywane są od lewej do prawej.

Znaczenie znaków „=” oraz „:=” w Pascalu

- a) „=” jest to **znak porównania** gdy porównujemy jakąś wartości w instrukcji np. IF , REPEAT,
- b) „:=” są to dwa znaki tzw. **przypisanie** stosowanie np. podczas pisania wzorów matematycznych.

WYRAŻENIA ARYTMETYCZNE

WYRAŻENIA ARYTMETYCZNE	ZAPIS W PASCAL
$W = \frac{A+B}{C \cdot D}$	W:=(A+B)/(C*D);
$L = 5 \sin \frac{A}{2} + 3 \cos TX$	L:=5*SIN(A/2)+3*COS(T*X);
$Y = \frac{A-B}{C^3}$	Y:=(A-B)/(SQR(C)*C);
$X_1 = \frac{-B + \sqrt{B^2 - 4 \cdot A \cdot C}}{2 \cdot A}$	X1:=(-B+SQRT(SQR(B)-4*A*C))/(2*A);
$S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$	S:=SQRT(P*(P-A)*(P-B)*(P-C));
$U = \frac{1}{2} \cdot E \cdot f^2$	U:=0.5*E*SQR(f);
$T = 2 \cdot \pi \cdot \sqrt{\frac{m}{k}}$	T:=2*PI*SQRT(m/k);
$C = \frac{1}{\log_4(x+1)}$	C:=LN(4)/LN(X+1);
$Y = \log(X^2 + 1)$	Y:=LN(SQR(X)+1)/LN(10);
$Z = 2 \cdot X^Y$	Z:=2*EXP(Y*LN(X));
$Y = 3 \cdot \operatorname{tg}(\alpha)$	Y:=3*SIN(ALFA)/COS(ALFA);

=====

Instrukcja **READ(zmienna lub zmienne).**

Znaczenie: wprowadzenie do zadeklarowanych zmiennych (komórek)wartości z klawiatury. Zmiennych w instrukcji. READ może być wiele, muszą być jednak tego samego typu. Zmienne muszą być zadeklarowane w bloku VAR. Kursor po wprowadzeniu danych pozostaje w linii, w której je wprowadziliśmy .

Instrukcja **READLN(zmienna lub zmienne).**

Znaczenie: takie same jak instrukcji READ lecz kursor przechodzi do nowej linii po wykonaniu instrukcji.

PRZYKŁAD (WPISZ DO KOMPUTERA I SPRAWDŹ DZIAŁANIE)

Program dodający dwie zmienne A i B a wynik zapisany jest w zmiennej C.

```
PROGRAM DOD;
USES
  CRT;
VAR
  A,B,C:INTEGER;
BEGIN
  A:=2;
  B:=3;
  C:=A+B;
  WRITELN('C=',C);
  REPEAT UNTIL KEYPRESSED;
END.
```

FORMATOWANIE WYDRUKU DANYCH

W celu lepszej czytelności wyprowadzanych danych stosowane jest *formatowanie wydruku danych*.

Polega to na tym, że dla liczb typu **INTEGER** po nazwie zmiennej oraz dwukropku piszemy szerokość matrycy np. WRITELN('a=',A:6);

Dla liczby typu **REAL** po zmiennej oraz dwukropku piszemy najpierw szerokość matrycy potem ilość miejsc po przecinku po dwukropku np. WRITELN('POLE=',POLE:7:4);.

Matryca jest szerokością wypisywania danej mierzonej w znakach:

- ◆ dla liczby **INTEGER** łącznie z minusem (jeśli istnieje).
- ◆ dla **REAL** łącznie z kropką i minusem.

PRZYKŁAD 3

(Wpisać do komputera treść programu i porównać ją z działaniem programu komentarzy w programie nie wpisuj)

```
PROGRAM WYDRUKI;
USES
  CRT;
VAR
  A,B:INTEGER;
  C:REAL;
BEGIN
  A:=100;
  B:=3;
  C:=A/B;
  WRITELN('A=',A:4);           { matryca 4 }
  WRITELN('A=',A:10);          { matryca 10 }
  WRITELN('B=',B:16);           { matryca 16 }
  WRITELN('C=',C:8:4);          { matryca 8, miejsc po przecinku 4 }
  WRITELN('C=',C:16:8);         { matryca 16, miejsc po przecinku 8 }
  REPEAT UNTIL KEYPRESSED;
END.
```

ZADANIE 4. (wpisz rozwiązanie do komputera)

Napisać program obliczający sumę odwrotności kwadratów dwóch liczy y i x dla y=4; x=5. Przed rozwiązaniem wykonaj schemat blokowy w zeszycie.

Zadeklaruj trzy zmienne (y, x, suma_odwrotnosci_kwad) jako liczby rzeczywiste. Dokonaj formatowania wyniku matryca sześć trzy miejsca po przecinku.

ZADANIE 5. (wpisz rozwiązanie do komputera)

Napisać program rozwiązujący równanie kwadratowe w postaci $ax^2 + bx + c = 0$

dla a=1; b=-5; c=6; czyli równanie $x^2 - 5x + 6 = 0$ Przed rozwiązaniem wykonaj schemat blokowy w zeszycie.

Do rozwiązania równania kwadratowego użyj wzoru na deltę (wyróżnik) oraz na pierwiastki:

$$\Delta = b^2 - 4ac$$

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a}$$

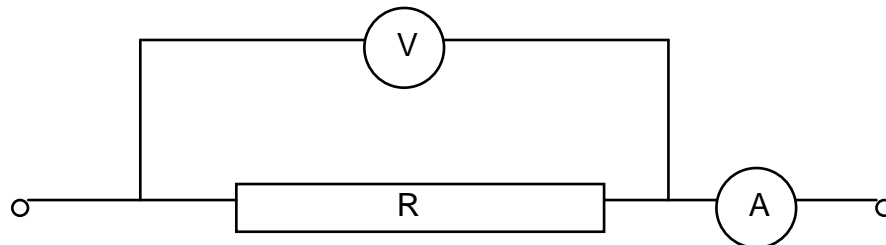
$$x_2 = \frac{-b + \sqrt{\Delta}}{2a}$$

(możesz zobaczyć jak zapisuje się wzory na x_1 i x_2 w PASCALU używając tabeli zaprezentowanej powyżej w tym rozdziale). Zadeklaruj sześć zmiennych (a , b , c , X_1 , X_2 , Δ) jako liczby rzeczywiste. Dokonaj formatowania wyników (X_1 , X_2 , Δ) matryca sześć dwa miejsca po przecinku.

PRZYKŁAD 4 (WPISĄĆ DO KOMPUTERA)

Napisać program obliczający wskazania amperomierza w następującym obwodzie.

gdzie: V – woltomierz
A – amperomierz
R – opór elektryczny
U – napięcie elektryczne czyli wskazanie woltomierza
I – prąd elektryczny czyli wskazanie amperomierza



prąd oblicz z prawa **Ohma** $I = U/R$

Dokonaj formatowania danych oraz wyników matryca osiem dwa miejsca po przecinku. W treści programu nie ma formatowania. Musisz dopisać je sam.

```
PROGRAM RAP;
  USES
  CRT;
  VAR
    U,I,R:REAL;
  BEGIN
    WRITE ('PODAJ U=');
    READLN(U);
    WRITE('PODAJ R=');
    READLN(R);
    WRITELN('DLA U=',U);
    WRITELN('DLA R=',R);
    I:=U/R;
    WRITELN('PRAD I=',I);
    REPEAT UNTIL KEYPRESSED;
  END.
```

ZADANIE 6. (ROZWIĄŻ I WPISZ DO KOMPUTERA)

Napisać program obliczający objętość dowolnego prostopadłościanu. Użyj formatowania wydruków. Bez sprawdzania poprawności wczytania danych.

ZADANIE 7. (ROZWIĄŻ I WPISZ DO KOMPUTERA)

Napisz program zamieniający stopnie na radiany. Program pyta się o ilość stopni i podaje ile to radianów.

$$x = \frac{s \cdot \pi}{180}$$

s – stopnie

x – radiany

Użyj formatowania wydruków. PI zapisz jako stałą 3.14159

ZADANIE 8. (ROZWIĄŻ I WPISZ DO KOMPUTERA)

Napisz program zamieniający radiany na stopnie. Program pyta się o ilość radianów i podaje ile to stopni. (Przekształć wzór z zadania poprzedniego). Użyj formatowania wydruków.

PRZYKŁAD 5 (WPISZ DO KOMPUTERA)

Obliczyć objętość walca. Przy wydrukach danych oraz wyników napisać jednostki, formatuj wydruki liczb. Gdy do wczytywania danych użyjesz jednej instrukcji READLN to możesz podać dane (tylko liczby) w jednej linii danych, liczby oddzielone są spacjami na koniec klawisz ENTER.

```
PROGRAM WALEC;
USES
  CRT;
CONST
  PI=3.1415;
VAR
  V,R,H:REAL;
BEGIN
  WRITELN('PODAJ R i H');
  READLN(R,H);      { jedna instrukcja wczytywania danych }
  V:=PI*R*R*H;
  WRITELN('DLA R=',R:6:2,' oraz H=',H:6:2);
  WRITELN('V=',V:6:2,' CM^3');
  REPEAT UNTIL KEYPRESSED;
END.
```

ZADANIE 9. (ROZWIĄŻ I WPISZ DO KOMPUTERA)

Napisz program obliczający przy stałej przeciwprostokątnej C=30 cm (wpisz jako stałą) i wczytywanej z klawiatury przyprostokątnej A:

- nieznaną bok trójkąta
- pole trójkąta
- cosinus, sinus, tangens dowolnego kąta ostrego
- kąty trójkąta wyrażone w stopniach

Przy wynikach pisz jednostki, formatuj wydruki.

Do obliczenia kątów użyj instrukcji ARCTAN(x) np. ALFA:=ARCTAN(1) to ALFA=PI/4 czyli ALFA:=45 stopni. Testuj program wykonując po jednym punkcie.

ZADANIE 10. (ROZWIĄŻ I WPISZ DO KOMPUTERA)

Obliczyć wartość wyrażenia

$$C = \frac{P(\operatorname{tg}(F1) - \operatorname{tg}(F2))}{W(U1 + U2)}$$

Dla danych P, F1, F2 (F1 i F2 w stopniach) W, U1, U2 wczytywanych z klawiatury. Do wczytania danych użyj jednej instrukcji READLN. W celu sprawdzenia poprawności programu wczytaj następujące dane

P=100 F1=45 F2=0 W=10 U1=1 U2=9 dla takich danych C w przybliżeniu powinno wyniesie 1.

Użyj formatowania wydruków.

Podczas rozwiązywania zadania pamiętaj, że:

- ♦ Pascal nie posiada funkcji tangens
- ♦ wartości funkcji trygonometrycznych obliczaj po zamianie kątów ze stopni na radiany
- ♦ pamiętaj o odpowiednich nawiasach przy zapisywaniu wzoru

PRZYKŁAD 6 (WPISZ DO KOMPUTERA)

Wczytać dwa punkty A(Xa, Ya) i B(Xb, Yb) do zmiennych Ax, Ay, Bx, By. Wydrukować wczytane punkty A i B np. gdy Ax=2,Ay=-3,Bx=6,By=1 w formacie 4 miejsca, jedno po przecinku. Wydruk powinien mieć postać A(2.0,-3.0) B(6.0,1.0) Obliczyć odległość między punktami A i B. Sformatuj również wydruk odległości. W programie wykonać komentarze, możesz bez polskich liter.

```
PROGRAM ODL;
USES
  CRT;
VAR
  Ax, Ay, Bx, By, D : REAL;
  { Ax, Ay- współrzędne punktu A }
  { Bx, By- współrzędne punktu B }
  { D - odległość punktów }
```

```

BEGIN
WRITELN('podaj punkt A');
READLN(Ax, Ay);
      { Współrzędne punktów wczytuj następująco
      pierwsza współrzędna spacja druga współrzędna ENTER }
WRITELN('Podaj punkt B');
READLN(Bx, By);      { wydruk wczytanych punktów }
WRITELN('A(',AX:4:1,',',AY:4:1,')'); {punkt A }
WRITELN('B(',BX:4:1,',',BY:4:1,')'); {punkt B }
D:=SQRT(SQR(Ax - Bx)+SQR(Ay - By));
WRITELN('D=',D:8:2);      { wydruk odległości }
REPEAT UNTIL KEYPRESSED;
END.

```

Uwaga: Przy rozwiązywaniu następnego zadania wykorzystaj treść tego przykładu poprzez kopiowanie i przerobienie treści przekładu.

ZADANIE 11. (ROZWIĄŻ I WPISZ DO KOMPUTERA)

Wczytać dwa punkty A(Xa, Ya) i B(Xb, Yb). Wydrukować wczytane punkty w formacie pięć miejsc, dwa po przecinku. Obliczyć współrzędne wektora a1 i a2 dla wektora $AB=[a1,a2]$ $AB=[Xb-Xa,Yb-Ya]$. Wydrukować wektor dla danych z poprzedniego zadania $AB=[4.0,4.0]$. Użyj komentarzy. W celu urozmaicenia prezentacji graficznej wyników programu użyj definiowania okien zmień kolor tła tego okna oraz kolor pisania , spróbuj zastosować inwersję oraz miganie (patrz program do prezentacji modułu CRT)

ZADANIE 12. (ROZWIĄŻ I WPISZ DO KOMPUTERA)

Treść zadania:

Oblicz kąt między wektorami. W tym celu:

Wykonaj czynności oraz obliczenia

- Wczytać punkty A B C D.
- Wypisz te punkty po wczytaniu np. A(2,-3).
- Obliczyć współrzędne wektorów AB i CD
- Wypisz te wektory po wczytaniu np. AB=[2,-3].
- Oblicz kąt między wektorami AB, CD. Podaj kąt w stopniach w formatach dogodnych dla użytkownika np. do jednego miejsca po przecinku.
- Użyj komentarzy. Zmień kolor tła tego okna oraz kolor pisania.

Wskazówka:

Kąt między wektorami obliczmy:

$$v1=[a1,b1] \quad v2=[a2,b2] \quad \cos_alfa:=(a1*a2+b1*b2)/(dv1*dv2)$$

dv1,dv2–długości wektorów. Długość wektora jest to pierwiastek z sumy kwadratów ich współrzędnych, czyli:

$$dv1 = \sqrt{a1^2 + b1^2}$$

$$dv2 = \sqrt{a2^2 + b2^2}$$

Następnie sin_alfa z jedynki trygonometrycznej potem tangens.

$$\sin_alfa = \sqrt{1 - (\cos_alfa)^2}$$

$$\tan_alfa = \frac{\sin_alfa}{\cos_alfa}$$

Do obliczenia kata użyj atan(tangens).

$$alfa_x = \arctan(\tan_alfa)$$

Z radianów zamień na stopnie.

$$alfa_s = \frac{alfa_x * 180}{\Pi}$$

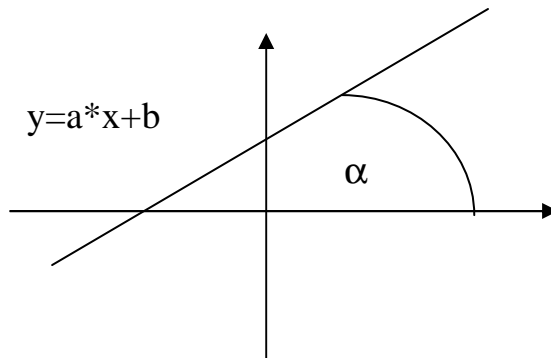
Użyj komentarzy. W celu urozmaicenia prezentacji graficznej wyników programu użyj definiowania okien zmień kolor tła tego okna oraz kolor pisania, spróbuj zastosować inwersję oraz miganie.

ZADANIE 13. (ROZWIĄŻ I WPISZ DO KOMPUTERA)

Treść zadania:

Oblicz wzór funkcji liniowej w postaci kierunkowej, czyli $y = a*x + b$ oraz podaj kąt, jaki tworzy ta prosta z osią X.

Po podaniu dwóch różnych punktów $A(x_A, y_A)$ oraz $B(x_B, y_B)$ przez które przechodzi ta prosta.



Rozwiązanie:

1) Oblicz współczynnik kierunkowy a ze wzoru

$$a = \frac{y_B - y_A}{x_B - x_A}$$

2) Oblicz współczynnik b ze wzoru

$$b = y_A - a * x_A$$

3) Wypisz prostą w postaci $y = a * x + b$ np. dla $a = 2$ i $b = 3$ $y = 2 * x + 3$

4) Oblicz kąt α w radianach ze wzoru

$$\alpha_x = \arctan(a)$$

5) Oblicz kąt w stopniach ze wzoru

$$\alpha_s = \frac{\alpha_x * 180}{\pi}$$

6) Wypisz α_s na ekran

Wprowadź $A(0,1)$ $B(1,2)$ a otrzymasz

$a = 1$ $b = 1$ $\alpha_s = 45$ stopni

SPRAWDZIAN 2 CZAS 40 MINUT. Przykładowy zestaw.

Uczeń wybiera zadanie na określoną ocenę

Zadanie 1 na ocenę dopuszczającą

Na startujący pojazd o masie m działa siła F . Oblicz prędkość V i przebytą drogę s po upływie czasu t od chwili startu. Użyj komentarze, wykonaj formatowanie wydruków oraz napisz jednostki.

$$V = F * t / m \quad s = F * t^2 / 2m$$

Zadanie 2 na ocenę dostateczną

Po wczytaniu do komputera ramienia A [cm] oraz podstawy trójkąta równoramiennego B [cm] oblicz pole oraz obwód trójkąta. Użyj komentarze, wykonaj formaty wydruków oraz zapisz jednostki.

Zadanie 3 na ocenę dobrą

Po wczytaniu do komputera tworzącej stożka L [cm] oraz kąta α [stopniach] zawartego między tworzącą a podstawą. Oblicz pole powierzchni bocznej stożka P oraz objętość V . Użyj komentarze, wykonaj formaty wydruków oraz zapisz jednostki.

Zadanie 4 na ocenę bardzo dobrą

Trzy punkty $A(x_1, y_1)$ $B(x_2, y_2)$ $C(x_3, y_3)$ tworzą trójkąt. Wczytać do komputera punkty A B C . Na tej podstawie oblicz pole trójkąta ABC . Wypisz wczytane punkty. Wzór na pole trójkąta $P = 1/2 * |x_1 * y_2 - x_2 * y_1 + x_2 * y_3 - x_3 * y_2 + x_3 * y_1 - x_1 * y_3|$. Oblicz oraz wypisz na ekranie monitora równanie prostej przechodzącej przez punkty AB w postaci $y = ax + b$. Użyj komentarze, wykonaj formaty wydruków oraz zapisz jednostki. W celu urozmaicenia prezentacji graficznej wyników programu użyj definiowania okien zmień kolor tła tego okna oraz kolor pisania, spróbuj zastosować inwersję oraz miganie.

CZEŚĆ TRZECIA WYKŁADU PASCALA

Teoretyczny opis zagadnień oraz instrukcji

Instrukcja złożona

Gdy chcesz pewien ciąg instrukcji połączyć w jeden blok to musisz użyć **instrukcji złożonej**. Słowo **BEGIN** pełni rolę nawiasu otwierającego, a słowo **END** pełni rolę zamykającego .

```
BEGIN           {początek instrukcji złożonej}
    instrukcja 1;
    instrukcja 2;
    .....
    .....
    instrukcja n;
END;           {początek instrukcji złożonej}
```

Instrukcja warunkowa (rodzaj→alternatywy)

Gdy program musi podjąć decyzję na podstawie warunku to należy użyć instrukcji alternatywy.

instrukcja pascala	polskie wytłumaczenie
IF warunek THEN	JESLI warunek WTEDY
instrukcja 1	instrukcja 1
{może być instrukcja złożona}	{może być instrukcja złożona}
ELSE	PRZECIWNIE
instrukcja 2	instrukcja 2
{może być instrukcja złożona}	{może być instrukcja złożona}

UWAGA! Przed słowem ELSE oraz po tym słowie nie ma średnika(;

Instrukcja warunkowa (rodzaj→wyboru, nie ma else)

IF warunek **THEN** instrukcja lub instrukcja złożona

Operatorv logiczne

Warunki w PASCALU można budować za pomocą operatorów logicznych:

OR AND NOT LUB I NIE

przykłady1:

IF (X>=1) OR (X<=-1) THEN { zapis przedziału x należy $(-\infty, -1 >$ lub $< 1, \infty)$ }

przykłady2:

IF (X>-1) AND (X<1) THEN	{ zapis przedziału x należy (-1,1) }
--------------------------	--------------------------------------

przykłady3:

REPEAT

.....

UNTIL ((ZNAK1='N') OR (ZNAK1='n'))AND((ZNAK2='P') OR (ZNAK2='p'));

uwaga: gdy występuje więcej niż jeden warunek logiczny zapisywane są one w nawiasach okrągłych.

Instrukcja REPEAT

Just to instrukcja z kontrolowanym **WYJŚCIEM**. Działanie tej instrukcji polega na tak długim wykonywaniu ciągu instrukcji dopóki wartość warunku nie jest spełniona.

REPEAT

```
instrukcja 1;
instrukcja 2;
    •
    •
    •
instrukcja n;
```

} powtarzający ciąg instrukcji aż do spełnienia warunku

UNTIL warunek:

Instrukcja WHILE

Jest to instrukcja z kontrolowanym **WEJŚCIEM**. Jej działanie polega na obliczaniu wartości warunku i dopiero wtedy wykonaniu instrukcji lub ciągu instrukcji.


```

WHILE warunek
DO
BEGIN
    instrukcja 1;
    instrukcja 2;
    •
    •
    •
    instrukcja n;
end;

```

wykonaj ciąg instrukcji po sprawdzeniu warunku

Przykład 7

Program rozpoznający znak liczby wprowadzanej z klawiatury. Rozpoznawanie polega na wczytaniu liczby z klawiatury oraz wydaniu jednego z komunikatów:

- ◆ liczba większa od zera
- ◆ liczba mniejsza od zera
- ◆ liczba równa zero.

Wykonaj w zeszycie schemat blokowy dla przykładu.

```

PROGRAM ZNAK;
USES
    CRT;
VAR
    X:REAL;
BEGIN
    WRITE ('PODAJ X=');READLN (X);
    IF X> 0 THEN
        WRITELN ('LICZBA WIĘKSZA OD ZERA')
    ELSE
        IF X<0 THEN
            WRITELN ('LICZBA MNIEJSZA OD ZERA')
        ELSE
            WRITELN ('LICZBA RÓWNA ZERO');
    REPEAT UNTIL KEYPRESSED;
END.

```

Przykład 8

Program, który będzie powtarzał napis " Informatyka " dopóki nie wciśniesz klawisza "n" lub "N".

Wykonaj w zeszycie schemat blokowy dla przykładu.

```

PROGRAM NAPIS;
USES
    CRT;
VAR
    ZNAK:CHAR;
BEGIN
    REPEAT
        WRITELN (' INFORMATYKA ');
        WRITELN ('Czy dalej t/n');
        READLN (ZNAK);
    UNTIL (ZNAK='N') OR (ZNAK='n');
END.

```

Zadanie 14.

Napisać program obliczający pole kwadratu. Sprawdź czy bok kwadratu jest większy od zera. Jeżeli jest większy oblicz pole jeśli nie to podaj komunikat "bok musi być większy od zera". W programie użyj **tylko jednej** instrukcji IF...THEN...ELSE. Zapewnij powtarzalność programu.

Wykonaj w zeszycie schemat blokowy dla zadania.

Zadanie 15.

Napisz program rozwiązujący równanie liniowe w postaci $A \cdot x = B$.

Równanie liniowe posiada trzy przypadki rozwiązania:

- jedno rozwiązanie (równanie oznaczone)

- równanie sprzeczne
- nieskończenie wiele rozwiązań (równanie nieoznaczone)

Warianty rozwiązania zależą od wartości liczb A i B . Zastanów się jakie są te zależności i zapisz specyfikację problemu oraz zapisz algorytm w postaci schemat blokowy.

Wykonaj program.

Zadanie 16.

Napisz program znajdujący największy wspólny dzielnik dla nieujemnych liczb całkowitych m i n algorytmem Euklidesa. Największy wspólny dzielnik oznaczamy symbolem $NWD(m,n)$.

Znajdź w zeszycie $NWD(\text{numer w dzienniku}+10, \text{miesiąc urodzenia}+10)$ dowolną metodą.

Zapisz w zeszycie:

Specyfikacja dla algorytmu Euklidesa.

Dane: Dwie liczby naturalne m i n , $m \leq n$.

Wynik: $NWD(m,n)$ – największy wspólny dzielnik m i n

Lista kroków dla algorytmu Euklidesa.

Krok 1 Jeśli $m=0$, to n jest szukanym dzielnikiem. Zakończ algorytm.

Krok 2 $r := (n \bmod m)$, $n := m$, $m := r$. Wróć do kroku 1

Wykonaj symulację znajdowania $NWD(14,21)$ w postaci uzupełnionej tabeli.

przejścia algorytmu	n	m	r
1	?	?	?
.....
.....

Wykonaj symulację znajdowania $NWD(1073,1517)$ w postaci uzupełnionej tabeli.

przejścia algorytmu	n	m	r
1	?	?	?
.....
.....

Zapisz algorytm w postaci schemat blokowy.

Wykonaj program.

Zadanie 17.

Napisz program rozwiązujący równanie kwadratowe w postaci $A \cdot x^2 + B \cdot x + C = 0$.

Równanie to rozpatrz dla przypadku liniowego ($A=0$) i kwadratowego ($A \neq 0$)

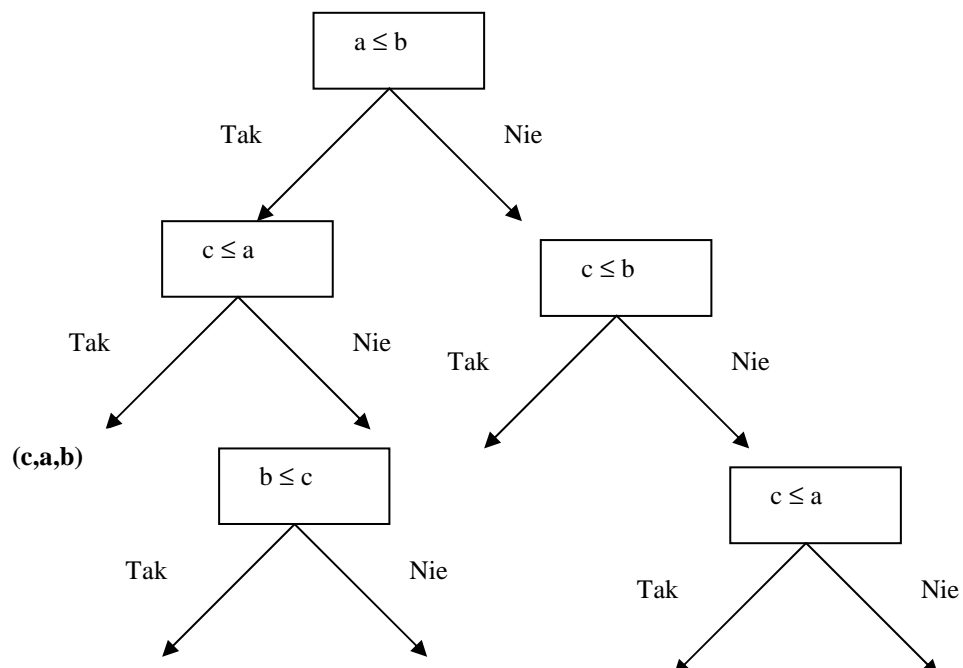
Zapisz specyfikację problemu oraz listę kroków. Zapisz algorytm w postaci schematu blokowego.

Wykonaj program.

Zadanie 18.

Napisz program porządkujący trzy liczby **a,b,c**.

Poniżej przedstawione jest drzewo algorytmu dla problemu porządkowania trójki liczb.



Drzewo posiada sześć możliwych dróg. Jedna z nich zakończona jest uporządkowaniem (c,a,b). Dopisz w zeszycie na drzewie pięć pozostałych uporządkowań.

W teorii algorytmów definiowane jest pojęcie **złożoności obliczeniowej** lub **pracochołoności algorytmu**.

Pod tym pojęciem będziemy rozumieli ilość porównań jakie wykonuje algorytm aby prawidłowo wykonać swoje działanie dla najbardziej pesymistycznego z punktu widzenia ilości obliczeń zestawu danych wejściowych do algorytmu.

Zapisz w zeszycie co to jest złożoność obliczeniowa oraz zapisz jaka jest ona dla zadania porządkowanie trójki liczb.

Zapisz w zeszycie specyfikację problemu.

Zapisz w zeszycie algorytm w postaci listy kroków.

Wykonaj program rozwiązujący problem porządkowania trójki liczb.

Zadanie 19.

Program, który przez wybór opcji podawanej z klawiatury obliczać będzie gdy:

1: pole trapezu

2: pole koła

Wpisz do komputera i dopisz wariant 2. Dopisz do programu jego powtarzalność.

PROGRAM WYBOR;

USES

CRT;

CONST

PI=3.1415;

VAR

S,H,R,A,B:REAL;

NRW:INTEGER; {NRW – numer wariantu}

BEGIN {początek programu głównego}

WRITELN ('Program umożliwia obliczanie');

WRITELN (' 1– pole trapezu');

WRITELN (' 2– pole koła');

WRITELN ('Podaj numer wariantu');

WRITE ('NRW=');READLN(NRW);

IF NRW=1 THEN

BEGIN {Początek instrukcji złożonej oraz wariantu 1}

WRITELN ('OBLICZANIE POLA TRAPEZU');

WRITELN ('Podaj wysokość, podstawę dolną i górną');

READLN (H,A,B);

S:= (A+B)*H*0.5;

WRITELN ('Pole S=',S:8:4);

END; {Koniec instrukcji złożonej oraz wariantu 1}

.....{tutaj dopisz wariant 2}

.....

.....

END. {Koniec programu}

Przykład 9

Program obliczający wartość funkcji danej wzorem

$$f(x) = \begin{cases} 1 & \text{dla } x \geq 1 \\ x * x & \text{dla } -1 < x < 1 \\ 1 & \text{dla } x \leq -1 \end{cases}$$

Dopisz do programu jego powtarzanie.

Posługując się programem wykonaj wykres funkcji f(x) w zeszycie. Najpierw wykonaj w zeszycie tabelę.

x						0					
y											

Uzupełnij pierwszy wiersz liczbami dwie > 1, dwie < -1, trzy z przedziału (-1,0), trzy z przedziału (0,1)

Wczytaj x a komputer obliczy y i tak otrzymany punkt zaznacz w układzie XY. Połącz punkty a otrzymasz wykres.

PROGRAM FUN;

USES

CRT

VAR

X,Y:REAL;

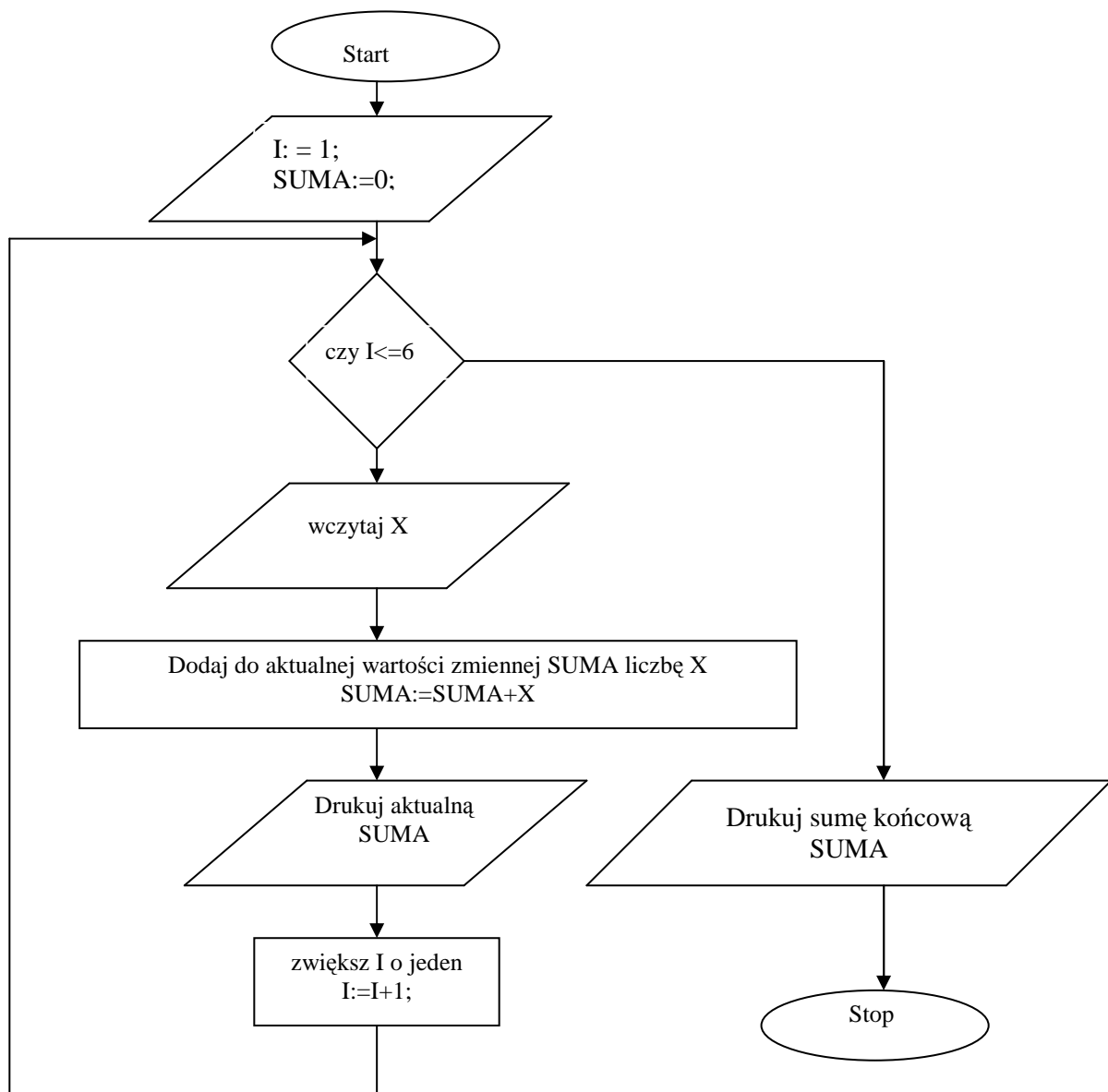
```

BEGIN
  WRITE ('Podaj x=');
  READLN (x);
  IF (X>=1) OR (X<=-1) THEN
    BEGIN { zapis przedziału x należy  $(-\infty, -1 >$  lub  $< 1, \infty)$  }
      Y:=1;
      WRITELN ('DLA X=',X:6:2,'   Y=',Y:6:2);
    END;
  IF (X>-1) AND (X<1) THEN
    BEGIN { zapis przedziału x należy  $(-1, 1)$  }
      Y:=X*X;
      WRITELN ('DLA X=',X:6:2,'   Y=',Y:6:2);
    END;
  END.

```

Przykład 10

Program, który będzie obliczał sumę sześciu liczb wczytanych z klawiatury. Komentarze możesz pominąć



```

PROGRAM SUMA1;
USES
  CRT
VAR
  I:INTEGER;
  SUMA:REAL;

```

```

{początek deklaracji zmiennych}
{I- licznik wczytanych liczb}
{SUMA-aktualna suma liczb}

```

```

X:REAL;           {wartość wczytanej liczby}
BEGIN             {początek części operacyjnej}
    I:=1;          {nadanie wartości początkowej zmiennej I}
    SUMA:=0;       {nadanie wartości początkowej zmiennej SUMA}
    WHILE I<=6 DO
        BEGIN     {początek instrukcji złożonej}
            WRITE ('Podaj X=');
            READLN(X);      {wczytanie liczby z klawiatury}
            SUMA:=SUMA+X;
            WRITELN ('Aktualna suma =',SUMA:6:2);
            I:=I+1;          {zwiększenie licznika wczytanych liczb}
        END;
        WRITELN ('Suma końcowa suma=',SUMA:6:2);
    REPEAT UNTIL KEYPRESSED;
END.

```

Zadanie 20.

Napisać program, który będzie można uruchomić przez podanie kodu dwu-literowego. Program obliczać będzie po podaniu długości boków gdy naciśniesz klawisz:

- 1.pole trójkąta
- 2.promień koła opisanego
- 3.promień koła wpisanego

Program po obliczeniu pyta się czy kończyć działanie opcje 1,2 lub 3. Program sprawdza poprawność wprowadzenia danych tzn. czy $A, B, C > 0$ oraz czy boki tworzą trójkąt. {jaki to warunek? }

Zadanie należy rozwiązywać w czterech etapach:

- 1.Zapewnić wejście do programu przez podanie szyfru. Komputer pyta się o pierwszą literę kodu, a następnie o drugą literę kodu następnie jeśli jest prawidłowy szyfr, to komputer drukuje "DZIEN DOBRY", jeśli jest zły "ZEGNAJ".
- 2.Komputer będzie wykonywał program (powtarzał napis DZIEN DOBRY) aż do wczytania liczby z przedziału $[-3, 2]$.
- 3.Komputer będzie wykonywał obliczenia właściwe.
- 4.Sprawdzenie poprawności danych. Gdy niewłaściwe dane to ich ponowne wprowadzenie.

Uwaga:

Po rozwiązaniu każdego etapu należy wpisać go do zeszytu lub wydrukować i wkleić do zeszytu tak aby mieć rozwiązane cztery etapy.

Wzory:

p– połowa obwodu
S– pole trójkąta
r– promień wpisany
R– promień opisany
A, B, C boki trójkąta

$$S = \sqrt{p(p - A)(p - B)(p - C)}$$

$$p = \frac{(A + B + C)}{2}$$

$$r = \frac{S}{p}$$

$$R = \frac{(ABC)}{4S}$$

Zadanie 21. (wykonanie tego zadania jest jednym z warunków aby otrzymać ocenę bardzo dobrą)

Napisz program porządkujący trzy liczby **a,b,c,d**.

Zapisz w zeszycie drzewo algorytmu dla problemu porządkowania czwórki liczb.

Zapisz w zeszycie złożoność obliczeniową dla tego problemu.

Zapisz w zeszycie specyfikację problemu.

Zapisz w zeszycie algorytm w postaci listy kroków.

Wykonaj program rozwiązujący problem porządkowania czwórki liczb.

Zadanie 22. (wykonanie tego zadania jest jednym z warunków aby otrzymać ocenę bardzo dobrą)

Napisz program rozwiązujący równanie dwukwadratowe w postaci $A*x*x*x*x+B*x*x+C=0$.

Równanie to rozpatrz dla przypadku liniowego, kwadratowego, dwukwadratowego.

Zapisz specyfikację problemu oraz listę kroków. Zapisz algorytm w postaci schematu blokowego.

Wykonaj program.

Jak uruchamiać oporne programy

Rodzaje błędów podczas testowania programów.

a)błędy kompilacji → powstają podczas fazy kompilacji programu np. błędy składni instrukcji.

b)błędy wykonania (ang. runtime errors)→ np. dzielenie przez zero czy próba otwarcia nieistniejącego pliku. Błędy te, zwane są znacznie mniej przyjemne i trudniejsze do usunięcia od błędów kompilacji.

Debugger → specjalne narzędzia uruchomieniowe, zwanego z angielska debuggerem (czyli odpluskwiaczem). Pozwala na wykryciu błędów wykonania.

Krokowe wykonywanie programu

- a) Trace Into → klawisz F7 polecenie Trace Into pozwala na "wejście" do wnętrza procedury,
- b) Step Over → klawisz F8 polecenie Step Over wykonuje procedurę ją jedną instrukcją.
- c) Reset → Ctrl-F2 pozwala na zrestartowanie programu (kasowanie niebieskiego paska).

Uwaga:

- 1) W obu przypadkach aktualnie wykonywana instrukcja zostaje wyróżniona w tekście programu kolorowym paskiem (niebieskim).
- 2) Tryb krokowy, pozwalający na ustalenie drogi, jaką "przebywa" wykonanie programu, nie daje żadnych wskazówek na temat wartości przyjmowanych przez zmienne, które w większości przypadków są odpowiedzialne za sterowanie pracą programu, a więc i ewentualne kolizje.

Podgląd wartości zmiennej używanej w programie po jego wykonaniu.

- d) Evaluate/modify → klawisze Ctrl-F4 z menu Debug. Podejrzenie zawartości wybranej zmiennej. Po wciśnięciu Ctrl+F4 zostanie wyświetlone okienko zawierające informację o wartości zmiennej oraz pozwalającego na jej zmianę (tak!).

-W pole **Expression** wpisujemy nazwę zmiennej, którego wartość chcemy obejrzeć,

-Wartość wyświetlana jest w polu **Result**,

-**New value** umożliwia jej zmianę.

Podgląd wartości zmiennych używanych w programie w czasie wykonania programu.

- e) Watch → klawisze Ctrl-F7. Po jego wydaniu i wpisaniu nazwy odpowiedniej zmiennej lub wyrażenia w okienku Add Watch na dole ekranu pojawi się okienko Watches, zawierające wartości śledzonych zmiennych.

-F7 → wykonanie krokowo programu z wyświetlaniem zmiennych w okienku Watch z wejściem do procedur,

-F8 → wykonanie krokowo programu z wyświetlaniem zmiennych w okienku Watch z wykonaniem procedur,

-klawisz Enter po najechaniu na nazwę zmiennej (Edit) → edycja zmiennej lub wyrażenia

-klawisz Insert dodanie nowej zmiennej w okienku Watch.

-klawisz Del kasowanie zmiennej w okienku Watch

-F10 górne menu.

Wykonanie programu do miejsca kursora w treści programu.

- f) Go to cursor → klawisz F4, powodujące wykonanie wszystkich instrukcji aż do miejsca wskazanego kursorem, a następnie przejście do pracy krokowej.

Wykonanie programu do określonego miejsca w treści programu.

- g) Add breakpoint z menu Debug → klawisz Ctrl-F8, znacznie bardziej użyteczne od funkcji Go to cursor okazuje się polecenie pozwalające na ustawienie w miejscu wskazanym kursorem tzw. **punktu wstrzymania**, powodującego zatrzymanie programu po każdorazowym jego osiągnięciu.

Dodatkowe parametry punktu wstrzymania:

-Condition → umożliwiają jego warunkowe wykonywanie,

-Pass count → zignorowanie określonej liczby przejść.

Punkt wstrzymania zostanie wyróżniona w treści programu kolorowym paskiem (czerwonym).

Po dojściu programu do punktu wstrzymania na ogół wystarczy sprawdzić zawartość podejrzaną zmienną poleceniem Evaluate lub Add Watch.

Kasowanie punktu wstawiania to ponowne wciśnięcie klawiszy Ctrl+F8.

Zadanie przykładowe na sprawdzian 3 czas 75 minut

Po wykonaniu każdego zadania zgłaszać nauczycielowi. Kolejność dowolna.

Zadanie1 (jeden punkt)

Program będzie uruchamiany przez podanie kolejno : numeru dnia urodzenia+10, pierwszej litery imienia oraz pierwszej litery nazwiska.

– Gdy podany jest prawidłowo szyfr komputer napisze "Brawo znasz moje parametry"

– Gdy szyfr jest nieprawidłowy komputer napisze "Nie znasz szyfru żegnaj".

Zadanie2 (jeden punkt)

Komputer zapyta czy chcesz powtarzać część programu drukującą "SPRAWDZIAN ZADANIE2". Jeśli tak to podaj liczbę z przedziału (–1,3> jeśli nie to pozostałe liczby.

Zadanie3 (jeden punkt)

Zostanie wyświetlone menu:

1– obliczanie objętości walca

2– obliczanie powierzchni całkowitej walca

Wybór wariantu 1 lub 2 wykonanie obliczeń oraz wyprowadzenie wyników na monitor z podaniem jednostek.

Zadanie4 (jeden punkt)

Sprawdzanie poprawności wczytanych danych z zadania 3

♠ numeru wariantu

♠ danych liczbowych

Zadanie5 (jeden punkt)

Po wczytaniu wartości dwóch kątów program poda wartość trzeciego kąta i odpowie jaki to jest trójkąt (prostokątny, równoboczny, równoramienny).

Zadanie6 (jeden punkt)

Napisz program porządkowania pięciu liczb bez użycia pętli i tablic.

ocena: punkty ocena

0–1 ndst	2 dopuszczający	3 dostateczny	4 dobry	5 bardzo dobry	6 celujący
-----------------	------------------------	----------------------	----------------	-----------------------	-------------------

CZEŚĆ CZWARTA WYKŁADU Z PASCALA

TABLICE → definicja

Tablica jest to struktura złożona z elementów tego samego typu.

Elementy tablicy są skazywane przez **indeks** lub **zespół indeksów**.

tablica jednowymiarowa

1	3	4	10
A[1]	A[2]	A[3]	A[4]

w komórce jest umieszczona A[1]=1 A[2]=3 A[3]=4 A[4]=10

PRZYKŁAD {nie wpisywać do komputera}

Program demonstruje sposób rezerwacji tablic jednowymiarowych

```
PROGRAM REZERWACJA_TABLIC;
VAR
  A:ARRAY[1..4] OF INTEGER; {rezerwacja tablicy A o czterech
                             elementach typu INTEGER}
  B:ARRAY[1..100] OF REAL; {rezerwacja tablicy B o stu
                             elementach typu REAL }
  C:ARRAY[1..8] OF CHAR; {rezerwacja tablicy C o ośmiu
                           elementach typu CHAR }
BEGIN
  WRITELN ('ten program tylko rezerwuje tablice jednowymiarowe');
END.
```

tablica dwuwymiarowa

A[1,1]	A[1,2]	A[1,3]	A[1,4]	A[1,5]	A[1,6]
3	1	2	10	6	5
1	3	4	20	7	9
A[2,1]	A[2,2]	A[2,3]	A[2,4]	A[2,5]	A[2,6]
pierwsza kolumna					szósta kolumna

czyli w komórce jest np.

A[1,1]=3 A[2,5]=7 A[1,3]=2 A[2,6]=9

miejsce w tablicy określone jest przez podanie nazwy tablicy i w nawiasach kwadratowych podajemy wiersz potem po przecinku kolumnę.

PRZYKŁAD {nie wpisywać do komputera}

Program demonstruje sposób rezerwacji tablic dwuwymiarowych

```
PROGRAM REZERWACJA_TABLIC;
VAR
  A:ARRAY[1..2,1..6] OF INTEGER; {rezerwacja tablicy A o dwóch
                                   wierszach i sześciu kolumnach typu INTEGER }
  B:ARRAY[1..3,1..3] OF REAL; {rezerwacja tablicy B o trzech
                                 wierszach i trzech kolumnach typu REAL }
BEGIN
  WRITELN ('ten program tylko rezerwuje tablice dwuwymiarowe');
END.
```

Instrukcja iteracyjna FOR

```
FOR I:=e1 TO e2 DO
  INSTRUKCJA;
I-zmienna kontrolna sterująca , licznik powtórzeń ( INTEGER )
```


e1–wartość początkowa zmiennej kontrolnej
e2–wartość końcowa zmiennej kontrolnej

Jeśli zamiast jednej instrukcji stosujemy instrukcję złożoną to
FOR I:=e1 TO e2 DO
BEGIN

INSTRUKCJA1;	}	→ <i>instrukcja złożona</i>
INSTRUKCJA2;		
.....		
INSTRUKCJAn;		

END;

Zapis ten oznacza, że instrukcja lub instrukcja złożona wykona się dla kolejnych wartości zmiennej kontrolnej I z krokiem 1 od wartości początkowej e1 do e2

FOR I:=e1 DOWNTO e2 DO
INSTRUKCJA;

I–zmienna kontrolna sterująca, licznik powtórzeń (INTEGER)
e1–wartość początkowa zmiennej kontrolnej
e2–wartość końcowa zmiennej kontrolnej

Jeśli zamiast jednej instrukcji stosujemy instrukcję złożoną to
FOR I:=e1 DOWNTO e2 DO
BEGIN

INSTRUKCJA1;	}	→ <i>instrukcja złożona</i>
INSTRUKCJA2;		
.....		
INSTRUKCJAn;		

END;

Zapis ten oznacza, że instrukcja lub instrukcja złożona wykona się dla kolejnych wartości zmiennej kontrolnej I z krokiem —1 od wartości początkowej e1 do e2.

Pętle

PRZYKŁAD 11

Program piszący na ekranie 100 wykrzykników

```
PROGRAM P3;
USES
  CRT;
VAR
  I:INTEGER;
BEGIN
  FOR I:=1 TO 100 DO
    WRITE('!');
  REPEAT UNTIL KEYPRESSED;
END.
```

Uwaga: Przy rozwiązywaniu następnego zadania wykorzystaj treść tego przykładu.

ZADANIE 23.

Napisać program drukujący 20 gwiazdek. Każda w nowej linii, wykorzystaj instrukcję FOR.

Uwaga: Przy rozwiązywaniu następnego zadania wykorzystaj treść tego przykładu.

ZADANIE 23a

Napisać program wypisujący liczby od numer w dzienniku+20 do numer w dzienniku. Każda liczba w nowej linii wykorzystaj instrukcję FOR ze zmniejszającą się wartością zmiennej sterującej. Wykonaj schemat blokowy

ZADANIE 24.

Napisać program drukujący liczby od 200 do 220 oraz ich pierwiastki. Wykorzystaj treść poprzedniego programu. Zastanów się jak liczba będzie początkiem pętli a jaka końcem oraz w jaki sposób uzyskać wydruk zadania jak poniżej.

Przykład wydruku zadania

SQRT(200)=14.14

.....

.....

SQRT(201)=14.18

ZADANIE 25.

Uwaga: Nagraj wszystkie etapy w osobnych plikach.

ETAP1

Napisać program drukujący na ekranie kody ASCII dla znaków o numerach od 33 do 255. Wydruk powinien mieć postać(przykład jednej linii) np.

kod znaku 33 !

.....

.....

Użyj pętli. W celu wydrukowania znaku o odpowiadającym mu kodzie użyj funkcji CHR(kod_znaku) np. WRITELN(CHR(65)); wyświetli literę A.

ETAP2

Zmodyfikuj program tak aby zatrzymał się jeśli zapisze cały ekran (np. 20 wierszy) i czekał, aż wciśniemy dowolny klawisz. Po wciśnięciu klawisza będzie drukował nowy ekran kodów. Do zmiany ekranów użyj instrukcji IF oraz MOD (sprawdź podzielność zmiennej sterującej I użytej w pętli FOR przez 20 z użyciem MOD →ponieważ użyjemy dwadzieścia wierszy na ekranie) gdy zmienna I jest podzielna przez 20 to zatrzymanie ekranu i oczekiwanie na naciśnięciu dowolnego klawisza (np. poprzez znak:=readkey;) czyszczenie ekranu i nowe 20 kodów ASCII.

ETAP3 (wykonują uczniowie, którzy chcą otrzymać ocenę bardzo dobrą za pracę na lekcji)

Zmodyfikuj program tak aby drukował na ekranie trzy słupki kodów i odpowiadających mu znaków.

PRZYKŁAD 12

Wpisz program rysujący linię pionową na środku ekranu składającą się z gwizdek. Następnie dopisz pętlę rysującą nową linię poziomą na wysokości 10.

program linia;

uses

crt;

var

i:integer;

begin

clrscr;

for i:=0 to 22 do

begin

gotoxy(40,i);

write('*');

end;

repeat until keypressed;

end.

Uwaga: Przy rozwiązywaniu następnego zadania wykorzystaj treść tego przykładu.

ZADANIE 26.

Zmodyfikuj przykład poprzedni tak aby gwiazdki rysowały się po przekątnej kwadratu o boku 20 linii ekranowych rozpoczynając od lewego górnego punktu ekranu.

ZADANIE 26a

Napisać program, który z użyciem czterech pętli (instrukcji iteracyjnej FOR) nasrysuje prostokąt z gwiazdek o następujących wymiarach

ilość wierszy= 5+(reszta z dzielenia numer_z_dziennika przez 3), ilość kolumn=11+(reszta z dzielenia numer_z_dziennika przez 2).

ZADANIE 26b

Napisać program, który po wczytaniu środka linii poziomej wykonanej ze znaków minus oraz długości tej linii narysuje tę linię.

Wygląd ekranu programu:
 Podaj zmieną (kolumnę ekranu) x=40
 Podaj zmieną (wiersz ekranu) y=10
 Podaj długość linii D=11
I teraz program rysuje tę linię.

ZADANIE 26c (wykonują uczniowie, którzy chcą otrzymać ocenę bardzo dobrą za pracę na lekcji)

Napisać program, zmodyfikuje zadanie poprzednie w taki sposób, ze nie przyjmie złych danych czyli:

x,y – z zakresu ekranowego okienka DOS

d - nie może być ujemne oraz musi być nieparzyste oraz takiej długości, że zmieści się na ekranie po uprzednim wczytaniu x i y.

PRZYKŁAD 13

Program rysujący z kropek połowę okręgu o równaniu

$$(x-24)^2 + y^2 = 24^2 \text{ po wyznaczeniu } y = \sqrt{576 - (x-24)^2}$$

otrzymujemy:

Jest to równanie okręgu o środku (24;0) i promieniu 24. Konieczne jest użycie instrukcji ROUND w celu zamiany zmiennej Y z typu REAL na INTEGER ponieważ instrukcja GOTOXY przyjmuje liczny INTEGER.

program okrag;

uses

crt;

var

i:integer;

begin

clrscr;

for i:=0 to 48 do

begin

gotoxy(i+1,1+round(sqrt(576-sqr(i-24))));

write('.');

end;

repeat until keypressed;

end.

ZADANIE 27. (wykonują uczniowie, którzy chcą otrzymać ocenę bardzo dobrą za pracę na lekcji)

Narysuj funkcję $y=12\sin(x)+12$ w zakresie jednego okresu $T=2*\pi$ czyli $\langle 0;6.28 \rangle$. W celu rozwiązania zadania powinienś dokonać przeliczenia zmiennej sterującej I która będzie się zmieniać w przedziale $\langle 0;79 \rangle$ (taki jest zakres ekranu dla X w trybie tekstowym) na zmienną X, która powinna zmieniać się w przedziale $\langle 0;6.28 \rangle$. Funkcja przeliczeniowa jest funkcją liniową w postaci $X=a*I+b$ gdzie szukanymi wielkościami są zmienne a i b:

gdy I=0 to X=0

gdy I=79 to X=6.28

aby obliczyć a i b postaw za I oraz X dane wielkości powyżej a otrzymasz układ równań z a i b rozwiąż go.

Utworzysz pętlę dla I od 0 do 79. Do instrukcji wpisz :

GOTOXY(I+1,ROUND(wzór funkcji, którą masz narysować ze wzorem przeliczeniowym jako argument));

Użycie funkcji ROUND jest konieczne ponieważ konieczna jest zamiana wartości real na integer ponieważ gotoxy przyjmuje tylko wartości integer. Do narysowania punktu użyj znaku . (kropka). Używając pętli narysuj oś X ze znaków — (minus) oś Y ze znaków | . Na końcach osi bez użycia pętli narysuj strzałki (> oś X oraz ^ oś Y).

ZADANIE 27a

Napisz program obliczający sumę liczb od numer_w_dzienniku+20 do numer_w_dzienniku+40. Wykonaj schemat blokowy.

ZADANIE 27b

Napisz program obliczający iloczyn liczb od miesiąc_urodzenia do miesiąc_urodzenia+3. Wykonaj schemat blokowy.

PRZYKŁAD 14

Polecenie:

Temat: Program do sprawdzania czy liczba wczytana z klawiatury jest liczbą pierwszą.

Wykonaj:

- 1) Wpisz w zeszycie temat programu.
- 2) Przepisz opis problemu.
- 3) Przepisz opis algorytmu.

Opis problemu:

Liczba pierwsza, liczba naturalna $n > 1$, dla której istnieją tylko dwa dzielniki naturalne: 1 i n . Największą znaną liczbą pierwszą jest $2^{69725932}-1$ (7 VII 1999), liczba ta zapisana w systemie dziesiętnym składa się z ponad 2 mln cyfr.

Opis algorytmu sprawdzania czy liczba jest liczbą pierwszą.

Sprawdzamy czy kolejne liczby naturalne od 2 do pierwiastek(n) są podzielnikami liczby n . Sprawdzenie podzielności odbywa się poprzez użycie funkcji MOD \rightarrow reszta z dzielenia, jeśli reszta z dzielenia jest zero to oznacza to, że liczba n jest podzielna przez liczbę mniejszą od n i n nie jest liczbą pierwszą. Należy zauważyć, że nie jest konieczne sprawdzanie kolejnych liczb naturalnych do 2 do n a wystarczy do pierwiastek(n). Zmienna k ma wartość domyślnie zero. Po zakończeniu programu sprawdzamy wartość zmiennej k , jeśli jest równa zero oznacza to, że liczba n jest pierwsza, jeśli nie (czyli jeden) to liczba nie jest pierwsza. Zmiana k zmieniana jest na wartość jeden gdy liczba n jest podzielna przez kolejną liczbę naturalną mniejszą od pierwiastka(n).

```
program liczba_pierwsza;
uses
  crt;
var
  k,i,n:integer;
begin
  clrscr;
  write('podaj liczbę n=');
  readln(n);
  k:=0;
  for i:=2 to round(sqrt(n)) do
    begin
      if (n mod i) =0 then
        begin
          k:=1;
          writeln(n, ' nie jest to liczba pierwsza bo dzieli sie przez ',i);
        end
      end;
  if k=0 then
    writeln(n, ' jest liczba pierwsza');
  repeat until keypressed;
end.
```

ZADANIE 27c

Wykorzystując poprzedni przykład napisz program, który wypisze liczby pierwsze z przedziału od miesiąc_urodzenia do miesiąc_urodzenia+50. Program wypisze również ile jest liczb pierwszych w tym przedziale. Wskazówki:

Poprzedni przykład umieść w pętli od miesiąc_urodzenia do miesiąc_urodzenia+50. Jeśli kolejna wartość zmiennej sterującej jest liczbą pierwszą to zostanie wyświetlona na ekranie oraz zostanie zwiększona zmienna LICZNIK o jeden. Pamiętaj o wyzerowaniu zmiennej LICZNIK przed wejściem do pętli.

Wygląd ekranu:

Podaj początek przedziału D_pocz=4

Podaj koniec przedziału D_kon=54

Liczby pierwsze

5

7

11

.....

.....

53

Liczba liczb pierwszych=16

ZADANIE 28.

W roku 1772 Leonard Euler podał wzór wielomianu o postaci:

$$w(i) = i^2 + i + 41$$

którego wartości dla $i = 0, 1, 2, \dots, 39$ są liczbami pierwszymi.

Napisz program znajdujący czterdzieści liczb pierwszych z użyciem pętli, korzystając z wzoru podanego przez Eulera.

Przykładowy sprawdzian**Zadanie1(1 punkt)**

Napisać program, który po wczytaniu dwóch liczb naturalnych większych od 1, czyli liczba **dol** oraz liczba **gor** gdzie $dol < gor$. Wypisze wszystkie liczby pierwsze bliźniacze. Liczby pierwsze bliźniacze to dwie liczby pierwsze różniące się między sobą o dwa np. 3 i 5.

Wygląd ekranu:

Podaj dolną liczbę przedziału przeszukiwania=1

Podaj górną liczbę przedziału przeszukiwania=20

Znalezione liczby bliźniacze to :

3 5

5 7

11 13

17 19

Zadanie2 (1 punkt)

Napisać program obliczając:

- sumę dwucyfrowych liczb naturalnych
- średnią dwucyfrowych liczb naturalnych
- ilość liczb dwucyfrowych podzielnych przez liczbę wczytaną z klawiatury.

Program powinien wypisać:

Podaj liczbę, której podzielność będziesz badał=

Suma to:.....

Średnia to:.....

Ilość liczb podzielnych przez..... to

Użyj pętli.

Zadanie3(1 punkt)

Wykonaj program rysujący linię pionową składającą się z podwójnej ilości gwiazdek na środku ekranu składającą się z gwiazdek. o wczytanej długości. Długość linii od 2 do 20.

np. dla $dlu=5$

```

**
**
**
**
**

```

Zadanie4(1 punkt)

Narysuj figurę jak poniżej z gwiazdek o długości i wysokości $liczba_liter_imienia+3$ użyciem dwóch pętli. Od miejsca ekranu (5,5).

```

*****
*
*
*
*
*
*

```

Zadanie5 (1 punkt)

Napisać program piszący 20 razy nazwisko ucznia. Każde w nowej linii, i przesunięte dwie kolumnę w lewo wykorzystaj instrukcję FOR. Początek pisania taki aby 20 napisów zmieściło się na ekranie.

np.

```

                Kowalski
            Kowalski
        Kowalski

```

Zadanie6 (1 punkt)

Napisać program drukujący liczby, ich kwadraty i sześciany od numeru w dzienniku do numeru w dzienniku+15

Przykład wydruku zadania (wygląd musi być identyczny)

```

I=4      I*I=16      I*I*I=64
I=5      I*I=25      I*I*I=125

```

```

.....
.....

```

ZADANIE 7

Liczby pierwsze w postaci $p, p+2, p+6, p+8$ nazywa się czworaczkami (np.: 101,103,107,109...). Nie wiemy, czy jest ich skończenie, czy nie skończenie wiele. Napisz program znajdujący czworaczki.

Tablice

Zadania teoretyczne o tablicach

Zadanie 1

Narysuj w zeszycie tablicę jednowymiarową o wymiarze 7. Zapisz do niej kolejne liczby nieparzyste naturalne, większe od Twojego numeru w dzienniku + 10 zaczynając od najmniejszej. Zapisz ile wynosi **a[2]** oraz **a[6]**. Zapisz instrukcję Pascal rezerwującą tę tablicę.

Zadanie 2

Narysuj w zeszycie tablicę jednowymiarową o wymiarze 2 na 5. Zapisz do niej kolejne liczby pierwsze, większe od Twojego numeru w dzienniku + 10 zaczynając od najmniejszej, tablicę zapisuj wierszami . Zapisz ile wynosi **a[1,2]** oraz **a[2,4]**. Zapisz instrukcję Pascal rezerwującą tę tablicę.

PRZYKŁAD 15

Program wpisujący do tablicy jednowymiarowej o 5 elementach dane wczytywane z klawiatury. Program drukuje również na monitorze zawartość tej tablicy. Stosuj wcięcia w programie.

```
PROGRAM P5;
  USES
  CRT;
  VAR
    I:INTEGER;  {zmienna sterująca}
    A:ARRAY[1..5] OF REAL;
BEGIN
  {wpisywanie danych z klawiatury}
  FOR I:=1 TO 5 DO
    BEGIN
      WRITE ('podaj A['I,']=');
      READLN (A[I]);
    END;
  {wypisanie zawartości tablicy}
  FOR I:=1 TO 5 DO
    BEGIN
      WRITELN('w komórce A['I,']=',A[I]:6:2);
    END;
  REPEAT UNTIL KEYPRESSED;
END.
```

Uwaga: Przy rozwiązywaniu następnego zadania wykorzystaj treść tego przykładu.

ZADANIE 29.

Napisać program nadający elementom tablicy jednowymiarowej o 10 elementach wartości w następujący sposób:

A[I]:=I np. A[2]:=2

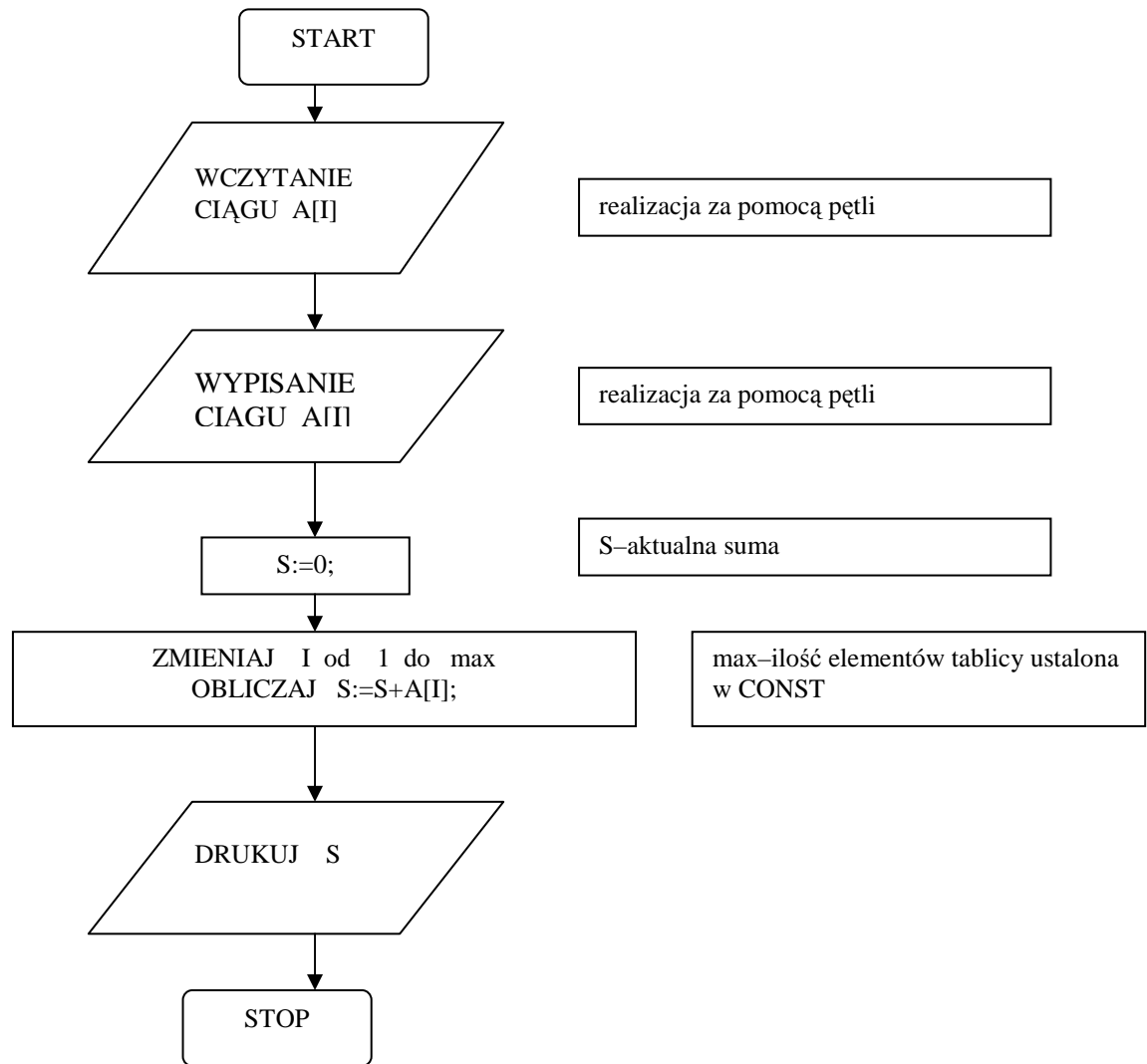
Wydrukuj zawartość otrzymanej tablicy. Wykorzystaj treść poprzedniego przykładu. Jeżeli komputer nadaje sam wartości elementom tablicy (nie są one wczytywane z klawiatury) to mówimy, że są one **generowane** przez komputer.

Uwaga: Przy rozwiązywaniu następnego zadania wykorzystaj treść tego zadania.

ZADANIE 30.

Napisać program obliczający sumę ciągu zapisanego w tablicy jednowymiarowej. Ilość wczytywanych elementów ciągu **max** wpisz jako stałą w bloku CONST. Program zrealizować w/g schematu blokowego.

Przyjmij max=7 tablicę w array zarezerwuj na 50 elementów. Wykonaj specyfikację problemu.

**ZADANIE 31.**

Zmodyfikuj tak program z zadania poprzedniego aby program liczył średnią arytmetyczną oraz geometryczną ciągu. Średnia geometryczna jest pierwiastek n -tego stopnia z wartości bezwzględnej iloczynu wyrazów ciągu. Pierwiastek n -tego stopnia z liczby jest inaczej podniesienie tej liczby do potęgi $1/n$. W jaki sposób podnosimy liczbę do dowolnej potęgi patrz część druga pascala.

ZADANIE 32.

Wczytać ciąg do tablicy. Wydrukować na ekranie wczytany ciąg. Oblicz następującą sumę

$$S = \sum_{n=1}^8 (1/x_n)$$

ZADANIE 33. (wykonują uczniowie, którzy chcą otrzymać ocenę bardzo dobrą za pracę na lekcji)

Napisz program do znajdowania elementu maksymalnego oraz minimalnego ciągu wczytanego z klawiatury (program będzie się pytał o długość ciągu) metodą **Min-i-Max**. Program powinien być tak napisany aby można było zobaczyć wczytany ciąg w jednym wierszu, w jednym wierszu ciąg kandydatów na max oraz w innym wierszu ciąg kandydatów na min oraz znaleziony element maksymalny oraz minimalny

PRZYKŁAD 16

Wczytać do tablicy o 3 wierszach i 5 kolumnach dane oraz wypisać jej zawartość na monitorze w celu sprawdzenia poprawności wczytywanych danych. Stosuj wcięcia w programie.

```

PROGRAM P6;
USES
  CRT;
VAR

```



```

      A:ARRAY[1..3,1..5] OF REAL;    { rezerwacja tablicy}
      I,J:INTEGER;    { deklaracja zmiennych sterujących}
BEGIN
    { wczytanie tablicy 3*5 }
    FOR I:=1 TO 3 DO
        BEGIN
            FOR J:=1 TO 5 DO
                BEGIN
                    WRITE('PODAJ A['',I','',J,']=');
                    READLN(A[I,J]);
                END;
            END;
        { wyprowadzenie tablicy A na monitor }
        FOR I:=1 TO 3 DO
            BEGIN
                FOR J:=1 TO 5 DO
                    WRITELN('A['',I','',J,']=',A[I,J]:6:2);
                END;
            REPEAT UNTIL KEYPRESSED;
        END.
    
```

Uwaga: Przy rozwiązywaniu następnego zadania wykorzystaj treść tego przykładu

ZADANIE 34.

Rozwiązać układ dwóch równań liniowych w postaci. Do rozwiązania użyj tablic i pętli.
np.

$$\begin{array}{rclcl}
 A[1,1] * X + A[1,2] * Y & = & A[1,3] & 2*X + 3*Y & = 5 \\
 A[2,1] * X + A[2,2] * Y & = & A[2,3] & -4*X + 8*Y & = 0 \\
 \text{i tak} & & A[1,1]=2 & A[1,2]=3 & A[1,3]=5 \\
 & & A[2,1]=-4 & A[2,2]=8 & A[2,3]=0
 \end{array}$$

A) gdy $W \neq 0$ układ ma jedno rozwiązanie

gdzie: $X = W_x / W$ $Y = W_y / W$

W —wyznacznik główny układu równań

W_x —wyznacznik dla zmiennej X

W_y —wyznacznik dla zmiennej Y

B) gdy ($W=0$) i (($W_x \neq 0$ lub $W_y \neq 0$)) układ sprzeczny

C) gdy ($W=0$) i ($W_x=0$) i ($W_y=0$) układ ma nieskończenie wiele rozwiązań sposób obliczania wyznaczników

Program pisz etapami :

$$W = \begin{vmatrix} A[1,1] & A[1,2] \\ A[2,1] & A[2,2] \end{vmatrix} = A[1,1] * A[2,2] - A[2,1] * A[1,2]$$

$$W_x = \begin{vmatrix} A[1,3] & A[1,2] \\ A[2,3] & A[2,2] \end{vmatrix} = A[1,3] * A[2,2] - A[2,3] * A[1,2]$$

$$W_y = \begin{vmatrix} A[1,1] & A[1,3] \\ A[2,1] & A[2,3] \end{vmatrix} = A[1,1] * A[2,3] - A[2,1] * A[1,3]$$

ETAP 1

Zapewnić wczytywanie danych do tablicy a oraz ich wypisywanie po wczytaniu z użyciem pętli.

ETAP 2

Obliczyć W_x , W_y , W , X , Y oraz wydrukować X i Y .

ETAP 3

Sprawdzenie warunków czy układ jest

- oznaczony
- sprzeczny
- ma nieskończenie wiele rozwiązań

wraz z wyprowadzeniem stosownego komunikatu na monitorze

ETAP 4

Zapewnienie powtarzalności obliczeń

ETAP 5

Wypisanie na monitorze układu równań w postaci np.

$$2*X + 3*Y = 5$$

$$-4*X + 8*Y = 0$$

Wykonaj specyfikację problemu.

PRZYKŁAD 17

Napisać program generowania tablicy 4x6 gdzie wartość wyrazu jest równa sumie indeksów tego wyrazu czyli :
 $a[i,j] := i + j$ czyli np. $a[2,3] := 2 + 3 = 5$

Wydrukuj tablicę na dwa sposoby:

- wyraz po wyrazie
- wierszami.

PROGRAM GENEROWANIE_TABLICY;

USES

CRT;

VAR

A:ARRAY[1..4,1..6] OF REAL; { rezerwacja tablicy }

I,J:INTEGER; { deklaracja zmiennych sterujących }

BEGIN

{ generowanie tablicy 4*6 }

FOR I:=1 TO 4 DO

BEGIN

FOR J:=1 TO 6 DO

BEGIN

A[I,J]:=I+J;

END;

END;

{ wyprowadzenie tablicy A na monitor WYRAZ PO WYRAZIE }

FOR I:=1 TO 4 DO

BEGIN

FOR J:=1 TO 6 DO

WRITELN('A[' , I , ',' , J , ']=', A[I,J]:6:2);

END;

{ wyprowadzenie tablicy A na monitor WIERSZAMI }

FOR I:=1 TO 4 DO

BEGIN

WRITE(A[I,1]:4:1, ' ', A[I,2]:4:1, ' ', A[I,3]:4:1);

WRITELN(' ', A[I,4]:4:1, ' ', A[I,5]:4:1, ' ', A[I,6]:4:1);

END;

REPEAT UNTIL KEYPRESSED;

END.

Uwaga: Przy rozwiązywaniu następnego zadania wykorzystaj treść tego przykładu.

ZADANIE 35.

Napisać program generowania tablicy 7*7 która będzie wyglądać następująco :

```

1 0 0 0 0 0 0      gdy i >= j to a[i,j]=1
1 1 0 0 0 0 0      i < j to a[i,j]=0
1 1 1 0 0 0 0
1 1 1 1 0 0 0
1 1 1 1 1 0 0
1 1 1 1 1 1 0
1 1 1 1 1 1 1

```

Wydrukuj tablicę dwoma sposobami wyraz po wyrazie oraz wierszami .

Wykonaj schemat blokowy.

ZADANIE 36.

Napisz program obliczający wyznacznik 3x3 metodą Sarussa. Wczytaj macierz oraz wydrukuj ją po wczytaniu wierszami oraz wartość wyznacznika.

ZADANIE 37. (wykonują uczniowie, którzy chcą otrzymać ocenę bardzo dobrą za pracę na lekcji)

Napisz program rozwiązujący układ trzech równań z trzema niewiadomymi metodą wyznacznikową. Uwzględnij przypadki jedno rozwiązanie, brak rozwiązań, nieskończenie wiele rozwiązań.

Wypisz postać układu równań oraz rozwiązanie.

ZADANIE 38.

Wczytać tablicę dwuwymiarową 2*5. Wydrukować tablicę wierszami. Znaleźć największy i najmniejszy element tej tablicy oraz wyprowadzić na ekran.

ZADANIE 39.

Napisz program, który po wczytaniu dwóch tablic A i B o wymiarze 4*3 wykona:

- wypisze obie macierze wierszami
- doda macierze i wynik wypisze wierszami
- odejmie macierze i wypisze wierszami
- pomnoży macierz A przez liczbę wczytana z klawiatury i wypisze wierszami.

Wykonaj schemat blokowy.

ZADANIE 40. (wykonują uczniowie, którzy chcą otrzymać ocenę bardzo dobrą za pracę na lekcji)

Napisać program, który będzie wczytywał tablicę, drukował ją na ekranie wierszami. Następnie pytał się o numery wierszy tablicy do zamiany. Jeśli podamy np. liczby 2 i 4 to zostaną zamienione wiersze drugi i czwarty.

Wydrukowana zostanie tablica wierszami po zamianie wierszy. Podczas wykonywania zadania konieczne będzie przedstawianie elementów tablicy można to osiągnąć w języku pascala następująco:

SKRYTKA[i]:=a[n,i];

a[n,i]:=a[m,i];

a[m,i]:=SKRYTKA[i];

gdzie SKRYTKA[i] jest tablicą pomocniczą a a[i,j] jest aktualnym elementem n, m numery wierszy do zamiany.

ZADANIE 41.

Wczytać ciąg do tablicy. Wydrukować na ekranie wczytany ciąg. Uporządkować od najmniejszego do największego wyrazy ciągu metodą bąbelkową. Wydrukować ciąg po sortowaniu.

Wykonaj listę kroków.

Zapisz w zeszycie przebiegi oraz zamieniane elementy sortowania metodą bąbelkową malejąco dla następującego ciągu:

a[1]=numer w dzienniku ; a[2]=miesiąc urodzenia; a[3]=dzień urodzenia; a[4]= numer w dzienniku + 40;

a[5]=miesiąc urodzenia

Sortowanie—oznacza proces porządkowania według pewnego klucza np.: od największego do najmniejszego, najmniejszego do największego, alfabetycznie itp.

Sortowanie może odbywać się według wielu różnych algorytmów, różniących się ilością wykonywanych operacji oraz szybkością działania

Metoda **bąbelkowa** polega: (porządkowanie od najmniejszego do największego—lub odwrotnie), ustawiamy się na pierwszym wyrazie ciągu i porównujemy go z drugim jeśli pierwszy jest większy od drugiego to zamieniamy je miejscami, następnie umieszczamy się na drugim i porównujemy z trzecim dokonując przestawienia lub nie.

Algorytm1

Porównując parami dochodzimy do końca ciągu (ustawiamy się na przedostatnim wyrazie). Jeśli podczas przechodzenia nastąpiła zmiana to musimy to zapamiętać jeśli nie to znaczy, że ciąg jest ustawiony prawidłowo. Po przejściu całego ciągu ustawiamy się ponownie na początek i sprawdzamy czy w poprzednim przechodzeniu było przestawienie jeśli nie było zamian to kończymy program jeśli było przestawienie to proces powtarzamy tyle razy aż nie będzie przestawienia.

Tablice uporządkować w porządku rosnącym.

9	2	7	10	8	4
---	---	---	----	---	---

przebieg porządkowania będzie następujący

pierwszy przebieg

zamienione elementy

krok 1 2 9 7 10 8 4

9 2

krok 2 2 7 9 10 8 4

9 7

krok 3 2 7 9 8 10 4

10 8

krok 4 2 7 9 8 4 10

10 4

drugi przebieg

zamienione elementy

krok 1 2 7 8 9 4 10

9 8

krok 2 2 7 8 4 9 10

9 4

trzeci przebieg

zamienione elementy

krok 1 2 7 4 8 9 10

8 4

czwarty przebieg

zamienione elementy

krok 1 2 4 7 8 9 10

7 4

w piątym przebiegu nie nastąpiła zmiana tzn., że można skończyć sortowanie.

Uwagi do rozwiązania zadania:

Uwaga1

Podczas porządkowania konieczne będzie przestawianie elementów tablicy można to osiągnąć w języku Pascala następująco:

```
SKRYTKA:=a[i];
```

```
a[i]:=a[i+1];
```

```
a[i+1]:=SKRYTKA;
```

gdzie SKRYTKA jest zmienną pomocniczą a a[i] jest aktualnym elementem.

Uwaga2

Konieczne będzie wprowadzenie zmiennej pomocniczej ZAMIANA typu integer. Możemy przyjąć, że zmienna ta przyjmuje wartość 0 gdy wchodzimy do przebiegu przez ciąg w celu sprawdzania czy elementy są ustawione w porządku od największego do najmniejszego. Sprawdzając parami elementy ciągu, i gdy napotykamy, że elementy nie są ustawione prawidłowo i trzeba je zamienić miejscami to zmienna ZAMIANA powinna przyjąć wartość 1. Wartość 1 jest informacją, że nastąpiło przestawienie i trzeba ponowić przejście przez ciąg i sprawdzania parami. Gdy po przejściu całego ciągu zmienna ZAMIANA ma wartość 0 to oznacza to, że ciąg jest uporządkowany i kończymy sortowanie.

Algorytm2

Algorytm sortowania bąbelkowego może być przedstawiane z użyciem listy kroków (jest on mniej efektywny niż algorytm1, ponieważ wykonuje większą ilość porównań). Poniższa lista kroków przedstawia sortowanie metodą bąbelkową.

KROK1: Dla $j = 1, 2, \dots, n-1$: wykonaj KROK2

KROK2: Dla $i = 1, 2, \dots, n-1$: jeśli $d[i] > d[i+1]$, to $d[i] \leftarrow d[i+1]$

KROK3: Zakończ algorytm.

Opis:

Zapis oznacza $d[i] \leftarrow d[i+1]$, że należy zamienić miejscami elementy $d[i]$ i $d[i+1]$. Zamianę można uzyskać poprzez:

```
SKRYTKA:=d[i];
```

```
d[i]:=d[i+1];
```

```
d[i+1]:=SKRYTKA;
```

gdzie SKRYTKA jest zmienną pomocniczą a d[i] jest aktualnym elementem.

ZADANIE 42. (wykonują uczniowie, którzy chcą otrzymać ocenę bardzo dobrą za pracę na lekcji)

Napisz program do sortowania ciągu wczytanego z klawiatury (program będzie się pytał o długość ciągu).

Gdy masz:

- numer parzysty w dzienniku to metodą wstawianie
- numer nieparzysty w dzienniku to metodą wybierania

Program powinien być tak napisany aby można było obserwować etapy wstawiania lub etapy wybierania.

Wykonaj listę kroków.

ZADANIE 43. (wykonują uczniowie, którzy chcą otrzymać ocenę bardzo dobrą za pracę na lekcji)

Treść zadania

Napisz program znajdujący wszystkie liczby pierwsze mniejsze lub równe od liczby n wczytanej z klawiatury metodą **sita Eratostenesa**.

Program powinien prezentować liczbę, której wielokrotności usuwamy ze zbioru oraz prezentować zbiór liczb, które pozostały oraz zbiór liczb odrzuconych i czekać na wciśnięcie dowolnego klawisza aby zaprezentować następne odrzucanie. Po znalezieniu liczb pierwszych wypisać ten zbiór i poinformować, że to koniec algorytmu.

Omówienie algorytmu rozwiązania problemu

W poprzednim zadaniu dotyczącym znajdowania liczb pierwszych znajdowaliśmy liczby pierwsze przez sprawdzanie podzielności. Sposób ten nie jest najefektywniejszą metodą wyszukiwania liczb pierwszych – dla każdego nowego kandydata musimy wykonywać określoną liczbę dzielen, tym większą, im większą wartość ma sprawdzana liczba. W czasach starożytnych znano już lepszy sposób, który opisał grecki uczony Eratostenes. Podszedł on do rozwiązania od drugiej strony – zamiast sprawdzać podzielność kolejnych liczb naturalnych przez znalezione liczby pierwsze, zaproponował on wyrzucanie ze zbioru liczb naturalnych wielokrotności kolejnych liczb (tych, które pozostały w zbiorze). To, co zostanie, będzie zbiorem liczb pierwszych. Metoda ta została nazwana **sitem Eratostenesa** i jest najszybszą metodą wyszukiwania liczb pierwszych w ograniczonym zbiorze.

Przykład

Zobaczmy jak działa sito Eratostenesa. Spróbujmy wg tej metody odszukać wszystkie liczby pierwsze w zbiorze 30 kolejnych liczb naturalnych.

{ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 }
--

Oto początkowy zbiór liczb większych od 1 i mniejszych od 31. Uwaga: liczba 1 – nie jest to liczba pierwsza,

ponieważ nie posiada dokładnie dwóch różnych dzielników.	
{ 2 3 5 7 9 11 13 15 17 19 21 23 25 27 29 }	→liczby, które pozostały
{ 4 6 8 10 12 14 16 18 20 22 24 26 28 30 }	→liczby usunięte
Bierzemy pierwszą liczbę 2 i usuwamy ze zbioru wszystkie jej wielokrotności.	
{ 2 3 5 7 11 13 17 19 23 25 29 }	→liczby, które pozostały
{ 9 15 21 27 }	→liczby usunięte
Następną liczbą jest 3. Usuwamy ze zbioru wszystkie wielokrotności liczby 3.	
{ 2 3 5 7 11 13 17 19 23 29 }	→liczby, które pozostały
{ 25 }	→liczby usunięte
Następną liczbą jest 5. Usuwamy ze zbioru wszystkie wielokrotności liczby 5.	
{ 2 3 5 7 11 13 17 19 23 29 }	
Zbiorze pozostały same liczby pierwsze.	

Pozostaje naturalne pytanie, do jakiej liczby pierwszej należy dojść, aby mieć pewność, iż reszta zbioru składa się wyłącznie z liczb pierwszych. Odpowiedź brzmi – do liczby pierwszej równej lub bezpośrednio mniejszej od pierwiastka z największej liczby w zbiorze wyjściowym. U nas największą liczbą było 30. Pierwiastek z 30 wynosi 5,47 – czyli do liczby pierwszej 5.

ZADANIE 44. (wykonują uczniowie, którzy chcą otrzymać ocenę celującą za pracę na lekcji)

Wygeneruj szesnaście początkowych liczb pierwszych z przedziału $\langle m;n \rangle$ wczytanego z klawiatury czyli wczytujemy m i n . Zapisz je w tablicy 4×4 i następnie wypisz tą tablicę wierszami. Oblicz sumę wyrazów nad główną przekątną (bez wyrazów przekątnej).

Wykonaj schemat blokowy.

ZADANIE 45. (wykonują uczniowie, którzy chcą otrzymać ocenę celującą za pracę na lekcji)

Napisz program na mnożenie macierzy.

Dane do programu:

–wymiary macierzy A i B

Program sprawdza czy mnożenie jest możliwe i teraz

–wczytujemy macierze A i B

Dane wyjściowe:

–macierz C wyprowadzana wierszami jako wynik mnożenia macierzy A i B .

Wykonaj specyfikację problemu oraz listę kroków.

PRZYKŁADOWE ZADANIA NA SPRAWDZIAN**ZADANIE 1**

Wczytać dane o czterech wektorach;

$u[1]=[a1,a2]$

$u[2]=[b1,b2]$

$u[3]=[c1,c2]$

$u[4]=[d1,d2]$

za pomocą instrukcji FOR...DO przyjmując tablicę danych w następujący sposób.

a1	b1	c1	d1
a2	b2	c2	d2

czyli np. $a[1,1]=a1$. Wypisać dane z tablicy. Wypisać tablicę wierszami.

ZADANIE 2

Dokonaj obliczeń długości wektorów $u[1]$ $u[2]$ $u[3]$ $u[4]$ (zapisz je w tablicy **u**) oraz wydrukuj te długości do dwóch miejsc po przecinku przy użyciu pętli. Długość wektorów oblicz bez stosowania pętli ze wzoru:

$$u[1] = \sqrt{(a_1)^2 + (a_2)^2} = \sqrt{(a[1,1])^2 + (a[2,1])^2}$$

Do obliczeń stosuj tę część wzoru, która zawiera określenia miejsca w tabeli czyli drugą jego część. Pozostałe wzory $u[2]$, $u[3]$, $u[4]$ wymyśl sam.

Dane do sprawdzenia obliczeń $a[1,1]=3$ i $a[2,1]=4$ to $u[1]=5$.

Komputer powinien rozstrzygnąć, który z wektorów jest najdłuższy i drukować wektory od najdłuższego do najkrótszego. Użyj dowolnej metody sortowania

ZADANIE 3

Wygenerować następującą tablicę używając pętli podwójnej.

1 0 0 0 0

0 4 0 0 0

0 0 9 0 0

0 0 0 16 0

0 0 0 0 25

oraz wydrukować na ekranie wierszami.

ZADANIE 4

Wczytać dwie macierze (tablice) **A** i **B** o wymiarze $n*2$ gdzie n wczytywane jest z klawiatury. Wypisz wierszami macierze **A** **B** **C**. Oblicz $C=2A-3B$. Następnie program powinien liczyć ilość wyrazów macierzy (tablicy) **C** większych od zera, równych zero oraz mniejszych od zera.

ZADANIE 5

Wczytaj dowolne dane do tablicy jednowymiarowej a_i i następnie oblicz następującą sumę

$$S = \sum_{i=3}^6 (2 \cdot a_i - 2)$$

ZADANIE 6

Oblicz rozpiętość ciągu ośmio elementowego wczytanego z klawiatury do tablicy jednowymiarowej. Wypisz ten ciąg wyraz po wyrazie. Rozpiętość ciągu to różnica między elementem maksymalnym a minimalnym.

CZĘŚĆ V WYKŁADU Z PASCALA PROCEDURY I FUNKCJE LOSOWA CASE ORAZ GRAFIKA

Uwaga:

Funkcja (procedura) przypomina strukturą mały program, toteż deklaracje zmiennych lokalnych umieszcza się zwykle na jej początku, po nagłówku.

PROCEDURY (PODPROGRAMY)

a) deklaracja procedury *jest umieszczana w programie po deklaracji zmiennych*

```
PROCEDURE nazwa_procedury(lista_zmiennych:typ);           {początek procedury}
CONST
    definicja stałych;
TYPE
    definiowanie typów programisty;
VAR
    definiowanie zmiennych;
DEKLARACJA PODPROGRAMÓW                                {procedur danej procedury}
BEGIN            {początek część operacyjnej procedury}
    instrukcja1;
    .....
    instrukcja_n;
END;                                                     {koniec procedury}
```

b) wywołanie procedury

NAZWA_PROCEDURY(lista zmiennych);

uwaga:

Mogą być procedury bez zmiennych (bezparametrowe) wtedy podajemy tylko
NAZWA_PROCEDURY

Rodzaje zmiennych ze względu na zasięg:

a) zmienne lokalne

Jak sama nazwa wskazuje, zmienna lokalna jest "prywatną własnością" funkcji lub procedury, wewnątrz której została zdefiniowana. Zmienna taka nie jest widoczna na zewnątrz funkcji (czyli np. w wywołującym ją programie), natomiast jest dostępna dla wszelkich funkcji i procedur zdefiniowanych w obrębie danej funkcji lub procedury (o ile ich definicje umieszczone są po deklaracji zmiennej). Zmienne lokalne umieszczane są z kolei na **stosie**, czyli w specjalnym obszarze pamięci przeznaczonym do tymczasowego przechowywania informacji. Zmienne lokalne są tworzone w momencie rozpoczęcia wykonywania funkcji i znikają po jej zakończeniu.

b) zmienna globalna

Zmienna globalna, "widoczna" w całym obszarze programu od momentu jej zadeklarowania. Każda zdefiniowana w programie funkcja i procedura ma pełny dostęp do wszystkich zmiennych globalnych. Zmienne globalne przechowywane są w tzw. segmencie danych i istnieją przez cały czas wykonywania programu.

Przysłanianie(maskowanie) zmiennych

Zjawisko to zachodzi gdy zmienna globalna ma identyczną nazwę jak zmienna lokalna nazywa.

Mechanizm ten zachodzi w Pascalu aby dać gwarancję, że nawet jeśli zadeklarujesz w funkcji zmienną lokalną o nazwie takiej samej jak zmienna zewnętrzna (w szczególności globalna), wszelkie odwołania wewnątrz funkcji będą zawsze kierowane do zmiennej lokalnej, zaś odpowiednia zmienna zewnętrzna pozostanie nienaruszona. Dzięki temu nie musisz dbać o to, by zmienne deklarowane w ramach poszczególnych funkcji miały różne nazwy i by któraś przypadkiem nie nazywała się tak samo, jak odpowiednia zmienna globalna.

Uwaga:

Wszystkie zmienne przeznaczone wyłącznie do użytku wewnętrznego funkcji i procedur powinny być bezwzględnie deklarowane jako zmienne lokalne. Zmienne globalne powinny być wykorzystywane tylko do przechowywania danych istniejących przez cały czas trwania programu.

Przekazywanie parametrów przez wartość lub przez nazwę (inaczej zmienną, wskaźnik, referencję)

a) przekazywanie przez wartość:

Procedure pole_obwod(p,q:integer; pole,obwod:real);

b)przekazywanie przez nazwę

Procedure pole_obwod (p,q:integer; **var** pole,obwod:real);

Kiedy stosować przekazywanie parametrów przez wartość, a kiedy przez nazwę? Przekazywanie przez wartość używane jest w "komunikacji jednokierunkowej", czyli wówczas, gdy zwrócenie wartości do wywołującego programu nie jest wymagane lub wręcz jest niewskazane (tj. chcemy zabezpieczyć się przed modyfikacją obiektu będącego parametrem). Jeżeli zachodzi potrzeba przekazania wartości z powrotem do wywołującego, konieczne jest użycie przekazywania przez nazwę, czyli poprzedzenie nazwy parametru słowem **var**.

PRZYKŁAD 18

Napisać program z użyciem procedur wyświetlający na monitorze wizytówkę. Przy wpisywanie komentarze możesz pominąć.

Uwaga1: Dotyczy wszystkich zadań oraz przykładów w całej instrukcji. Po uruchomieniu programu czy przykładu pojawi się nazwisko i imię ucznia następnie po naciśnięciu klawisza ENTER zostanie skasowany ekran i uruchomi się program czy przykład.

wpisanie przykładu 1 i nagranie go na dyskietkę ucznia pod nazwą cztery pierwsze litery nazwiska (nie stosuj polskich liter) podkreślenie numer przykładu np. kowa_p1

Uwaga2:Wszystkie przykłady oraz zadania powinny być nagrane na dyskietce ucznia pod nazwą cztery pierwsze litery nazwiska (nie stosuj polskich liter) podkreślenie numer zadania np. kowa_z1.pas lub kowa_p1.pas oraz dokonać kompilacji na EXE i zapisać na dysk pod nazwami np. kowa_z1.exe .

PROGRAM WIZYTOWKA;

USES

CRT;

VAR

DALEJ:CHAR;

{teraz deklaracja PROCEDUR}

PROCEDURE GWIAZDKI;{ta procedura nie posiada zmiennych}

BEGIN

WRITELN('*****');

END;

PROCEDURE NAZWISKO;

BEGIN

WRITELN('tutaj wpisz swoje nazwisko');

END;

PROCEDURE IMIE;

BEGIN

WRITELN('tutaj wpisz swoje imię');

END;

BEGIN {początek programu głównego}

CLRSCR; {czyszczenie ekranu}

{teraz wywołanie PROCEDUR bez zmiennych}

GWIAZDKI;

GWIAZDKI;

GWIAZDKI;

IMIE;

NAZWISKO;

GWIAZDKI;

GWIAZDKI;

GWIAZDKI;

DALEJ:=READKEY; {zatrzymanie programu aż do wciśnięcia klawisza}

END. {koniec programu}

ZADANIE 46.

Napisać program z użyciem procedur wyświetlający na monitorze trzy prostokąty oraz trzy trójkąty konstruowane z znaków "*". Zdefiniuj procedury PROSTOKAT oraz TROJKAT , a następnie wywołaj trzy razy te procedury.

Drukuj trójkąty i prostokąty naprzemian.

PRZYKŁAD 19

Napisać program z użyciem podprogramu , który będzie rysować prostokąt w żądanym miejscu ekranu. Linie poziome składać się będą ze znaków "-" a pionowe "!" . Prostokąt będzie rysował się zawsze o tej samej wielkości. Po wpisaniu programu zmodyfikuj jego treść tak aby narysowały się trzy prostokąty w różnych miejscach ekranu.


```

PROGRAM RYS;
  USES
    CRT;
  VAR
    DALEJ:CHAR;
{teraz deklaracja PROCEDURY}
{zmienne X,Y są to współrzędne górnego lewego rogu prostokąta}
  PROCEDURE PROST(X,Y:INTEGER);{ta procedura posiada zmienne}
  BEGIN
    GOTOXY(X,Y);
    WRITELN('—————');
    GOTOXY(X,Y+1);
    WRITELN('!      !');
    GOTOXY(X,Y+2);
    WRITELN('!      !');
    GOTOXY(X,Y+3);
    WRITELN('!      !');
    GOTOXY(X,Y+4);
    WRITELN('—————');
  END;

BEGIN {początek programu głównego}
  CLRSCR; {czyszczenie ekranu}
  {teraz wywołanie PROCEDUR ze zmiennymi X=5 i Y=6}
  PROST(5,6);
  DALEJ:=READKEY; {zatrzymanie programu aż do wciśnięcia klawisza}
END. {koniec programu}

```

Przykład 19a

Demonstracja przekazywania parametrów przez wartość oraz przez nazwę.

Program oblicza obwód i pole rombu po podaniu przekątnych (p,q).

1)Wpisz przykład do komputera i nagraj pod nazwą P19a_cztery_pierwsze litery nazwiska.pas

2)Zapisz w zeszycie słowa „Przekazywanie parametrów przez wartość” a pod tymi słowami z ekranu przepisz, jaki był wynik działania programu.

3)Zmień treść programu, aby przekazywanie było przez nazwę.

4)Zapisz w zeszycie słowa „Przekazywanie parametrów przez nazwę” a pod tymi słowami z ekranu przepisz, jaki był wynik działania programu.

5)Przepisz poprawiony przykład do zeszytu.

Program demonstracja_przekazywania_parametrow;

```

Uses
  crt;
var
  obwod1,pole1:real;
Procedure pole_obwod(p,q:integer; pole,obwod:real);
begin
  pole:=(p*q)/2;
  obwod:=2*sqrt(p*p+q*q);
end;

begin
  clrscr;
  pole_obwod(12,16,obwod1,pole1);
  writeln('obwod=',obwod1:6:2,' pole=',pole1:6:2);
  repeat until keypressed;
end.

```

ZADANIE 47.

Napisać program rysujący prostokąt w określonym miejscu ekranu oraz o żądanej długości i wysokości. Zdefiniuj procedurę PROST z czterema parametrami:

X,Y – współrzędne górnego lewego rogu prostokąta

A,B – szerokość i długość prostokąta

Linie poziome składają się ze znaków "-"

Linie pionowe składają się ze znaków "!"

Program pisz etapami, każdy etap testuj.

Etap1

napisz procedurę LINIAPOZIOMA z trzema parametrami X,Y,A wywołaj tę procedurę w celu sprawdzenia działania

Etap2

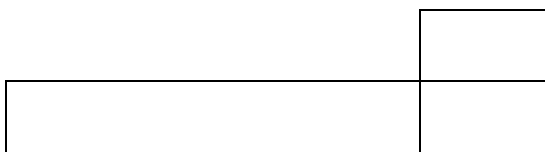
dopisz procedurę PROST z czterema parametrami X,Y,A,B. Procedura ta wywoła dwie procedury LINIAPOZIOMA z takimi parametrami, że będą się rysowały dwie linie poziome o długości równej szerokości i oddalone od siebie o wysokość prostokąta. wywołaj tę procedurę w celu sprawdzenia działania

Etap3

dopisz procedurę LINIAPIONOWA z trzema parametrami X, Y, B wywołaj tę procedurę w celu sprawdzenia działania

Etap4

dopisz dwukrotne wywołanie procedury LINIAPIONOWA w procedurze PROST w taki sposób aby powstał prostokąt. Poprzez wywołanie procedury PROST z różnymi parametrami otrzymaj dwa różne prostokąty (różne wymiary) stykające się bokami np.



FUNKCJE

a) deklaracja funkcji (definiowanie) – jest umieszczana w programie po deklaracji zmiennych

FUNCTION nazwa_funkcji(lista_zmiennych:typ):typ_funkcji; {początek}

CONST

definicja stałych;

TYPE

definiowanie typów programisty;

VAR

definiowanie zmiennych lokalnych dla funkcji;

DEKLARACJA PODPROGRAMÓW(procedur danej funkcji)

BEGIN {początek część operacyjnej funkcji}

co najmniej jedna instrukcja przypisania,

której lewą stroną jest nazwa funkcji

END; {koniec funkcji}

b) wywołanie funkcji

podanie nazwy funkcji z określonymi zmiennymi zmiennych

PRZYKŁAD 20

Program definiuje funkcję obliczającą przekątną kwadratu gdzie zmienną jest bok kwadratu. Następnie demonstrowane jest wywołanie tej funkcji w dwojaki sposób:

- pierwszy sposób to wywołanie dla ściśle określonego boku
- drugi sposób to wywołanie dla wartości wczytywanych z klawiatury

PROGRAM PRZEKATNA;

USES

CRT;

VAR

A1,D:REAL;

DALEJ:CHAR;

FUNCTION PRZEK(A:REAL):REAL;

BEGIN

PRZEK:=A*SQRT(2);

END;

BEGIN

CLRSCR;

```

WRITELN('NAJPIERW OBLICZENIA DLA A=100');
D:=PRZEK(100);
WRITELN('D=',D:6:2);
WRITELN('OBLICZENIA DLA WCZYTYWANYCH A Z Klawiatury');
WRITE('PODAJ A=');READLN(A1);
D:=PRZEK(A1);
WRITELN('DLA A=',A1:6:2,' D=',D:6:2);
DALEJ:=READKEY;
END.

```

ZADANIE 48.

Napisać program obliczający pole powierzchni całkowitej oraz objętość walca. W tym celu zdefiniuj funkcje POW oraz OBJ ze zmiennymi R i H. Oblicz pole oraz objętość dla

- a) R=10 oraz H=30 (ściśle określone)
- b) dla dowolnych wczytanych R i H wczytanych z klawiatury

PRZYKŁAD 21

Dokonaj tablicowania funkcję $f(x) = \cos^2(x) + \sin(x) \cdot |\sin(x)|$ w przedziale $\langle -5, 5 \rangle$ z krokiem 0.5.

Tablicowanie funkcji jest czynnością, która polega na obliczeniu wartości funkcji y dla kolejnych argumentów funkcji x. Krok tablicowania jest to liczba o jaką wartość będzie wzrastać wartość argumentu x. Komentarze możesz pominąć. Treść tego przykładu wykorzystaj w następnym zadaniu.

```

PROGRAM TABLICA;
USES
  CRT;
CONST
  KROK=0.5;
VAR
  X,Y:REAL;
  DALEJ:CHAR;
FUNCTION F1(X:REAL):REAL;
BEGIN
  F1:=sqr(cos(x))+sin(x)*abs(sin(x));
END;
BEGIN
  CLRSCR;
  X:=-5; {NADANIE WARTOSCI POZATKOWEJ ZMIENNEJ X}
  WHILE X<=5 DO
  BEGIN
    Y:=F1(X); {OBLICZENIE WARTOSCI FUNKCJI}
    WRITELN('f(',X:4:1,',')= ',Y:6:2);
    X=X+KROK; {ZWIĘKSZENIE ARGUMENTU O KROK=0.5}
  END;
  DALEJ:=READKEY;
END.

```

ZADANIE 49.

Do rozwiązywania tego zadania wykorzystaj treść przykładu poprzedniego. Ztablicować funkcję z krokiem 0.2 określoną następująco:

$$f(x) = \begin{cases} -1 & \text{dla } x \in \langle -2, -\frac{\pi}{2} \rangle \\ \sin(x) & \text{dla } x \in \langle -\frac{\pi}{2}, 0 \rangle \\ x * x & \text{dla } x \in \langle 0, 1 \rangle \\ \frac{1}{x} & \text{dla } x \in \langle 1, 2 \rangle \end{cases}$$

Funkcja jest definiowana przedziałami dlatego konieczne będzie użycie instrukcji IF podczas definiowania funkcji.

ZADANIE 50. (na ocenę bardzo dobrą)

Napisać program do obliczania symbolu Newtona. Program wykonaj etapami:

Etap1

Napisać program z użyciem funkcji, która obliczać będzie wartość funkcji SILNIA. np. $4! = 1 * 2 * 3 * 4 = 24$. Funkcja SILNIA będzie miała zmienną N (integer) wartość funkcji będzie real.

Etap2

Po przetestowaniu funkcji SILNIA napisać funkcję NEWTON z dwiema zmiennymi N i K(integer) wynik rzeczywisty, (NEWTON—symbol newtona). Funkcja NEWTON będzie wykorzystywać funkcję SILNIA.

Program powinien sprawdzać poprawność wczytywania danych.

GRAFIKA

Wszystkie poprzednie programy i przykłady pisane były w trybie tekstowym. W celu pisania programów w trybie graficznym musisz wykonać kilka zabiegów takich jak np. uruchomienie grafiki. Tryb graficzny jest to taki sposób pisania programów, że możesz zapalać/gasić pojedynczy pixel ekranu. Do rysowania oraz pisania w trybie graficznym służą inne instrukcje niż były w trybie tekstowym. Spis tych komend oraz jak ich użycie znajdziesz na końcu rozdziału.

Arc(x,y, 0,180-kat,R)-łuk, wycinek okręgu

Bar(x1,y1,x2,y2)-prostokąt „słupek”x1y1-lewy górny wierzch. Prost.,x2y2-prawy dolny,

Bar3d—trójwymiarowy słupek

Circle(X,Y,promień)-rysuje koło o wsp. xy i promieniu R

Ellipse(x,y,0,360,Rx,Ry)- rysuj elipsę

FillEllipse(x,y,kolor) –wypełnij elipsę

FillPoly- rysuj wielokąt wypełniony wewnątrz

FloodFill(X,Y,color_krawędzi)-wypełnij obszar od punktu XY

GetBkColor-podaj bieżący kolor tła

GetColor-podaj bieżący kolor rysowania

GetDefaultPalette—ustaw domyślną paletę kolorów

GetMaxX-podaj max szerokość

GetMaxY-podaj max wysokość

GetPixel-podaj kolor wskazanego punktu

Initgraph-inicjuje tryb graf

MoveTo(x,y)-przesuwa kursor graficzny o pozycje xy

MoveRel(dx,dy)-przesuwa kursor o dx,dy

Line(x1,y1,x2,y2)-rysuj linię prostą od x1,y1 do x2,y2

LineRel(dx,dy)-rysuje linie od wskazanego punktu

LineTo(x,y)-kreśli linię od położenia kursora graf do xy

OutText(x,y,tekst)-wprowadz text

OutTextXY-wprowadz text od współrzędnych xy

PutPixel(x,y,color)-stawia kropkę we wsp xy o kolorze

Rectangle(x1,y1,x2,y2)-rysuj prostokąt x1y1-lewy górny wierzch,x2y2-prawy dolny

SetBkColor(kolor1-15)-ustaw kolor tła

SetColor(kolor)-ustaw kolor rysowania

SetFillPatern-ustaw kolor wypełniacza figur

SetFillStyle(nr_wzorka,kolor)-ustaw rodzaj linii

SetGraphMode-włącz tryb graficzny

SetLineStyle(styl,wzorek,grubość)-ustawia rodzaj linii

SetWriteMode-narysowac na poprzedni rysunek

SetTextStyle(czcionka1-4,kierunek:horizdir-pozियोmo,vertdir-pionowo,rozmiar)-ustala rodzaj wpisywanego tekstu

TextHeight-ustaw wysokość tekstu

TexttWidth-ustaw szerokość tekstu

PRZYKŁAD 22

Narysować czerwony i wypełniony kwadrat w trybie graficznym oraz podpisać jaka to figura. Komentarze możesz pominąć. Treść tego przykładu wykorzystaj w następnym zadaniu.

```
PROGRAM KWAD1;
```

```
  USES
```

```
    CRT,GRAPH;
```

```
  VAR
```

```
    STEROWNIK,TRYB:INTEGER;
```

```
  BEGIN
```

```
    STEROWNIK:=DETECT;{ AUTOMATYCZNE USTALENIE RODZAJU KARTY }
```

```
      { GRAFICZNEJ PRZEZ KOMPUTER }
```

```
    INITGRAPH(STEROWNIK,TRYB,'C:\UCZEN\TP\BGI');
```

```

      {W APOSTROFACH PODAJEMY GDZIE MAMY NAGRANE}
      {STEROWNIKI GRAFICZNE DLA KARTY GRAFICZNEJ}
SETCOLOR(RED); {OKRESLENIE KOLORU RYSOWANIA}
SETLINESTYLE(0,3,3); {OKRESLENIE SPOSOBU RYSOWANIA LINII}
LINE(10,10,110,10); {RYSOWANIE LINII}
LINE(110,10,110,110);
LINE(110,110,10,110);
LINE(10,110,10,10);
SETFILLSTYLE(1,RED); {OKRESLENIE SPOSOBU RYSOWANIA WYPELNIENIA}
FLOODFILL(50,50,RED); {WYPELNIENIE OBSZARU}
SETCOLOR(GREEN);
SETTEXTSTYLE(0,0,0);
OUTTEXTXY(150,150,'KWADRAT');
REPEAT UNTIL KEYPRESSED;
END.

```

ZADANIE 51.

Narysować w trybie graficznym wypełniony trójkąt równoboczny o boku $200+5 \cdot (\text{numer z dziennika})$ pikseli oraz oznaczyć jego wierzchołki literami A B C.

Kolor wypełnienia:

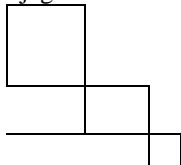
miesiąc urodzenia	kolor
styczeń, luty	zielony
marzec, kwiecień	brązowy
maj, czerwiec	żółty
lipiec, sierpień	niebieski
wrzesień, październik	turkusowy
listopad, grudzień	karmazynowy

Nazwy oraz ich numery szukaj w części pierwszej w przykładowym programie modułu CRT.

W zeszycie narysuj układ współrzędnych (tylko jedna ćwiartka). Pamiętaj, że punkt (0,0) jest w górnym lewym rogu układu (ekranu) i strzałki osi będą skierowane na dół. Narysuj trójkąt. Obok wierzchołków zapisz ich współrzędne. Do określenia współrzędnych wierzchołków możesz posłużyć się wzorem na wysokość w trójkącie równobocznym. Treść poprzedniego przykładu wykorzystaj w zadaniu.

PRZYKŁAD 23

Wykorzystując instrukcję graficzną LINE(X1,Y1,X2,Y2) z czterema parametrami X1,Y1 – współrzędne początku odcinka oraz X2,Y2 współrzędne końca odcinka zdefiniować procedurę KWADRAT(X,Y,D) X,Y – współrzędne lewego górnego rogu kwadratu D – długość boku kwadratu. Komentarze możesz pominąć. Treść tego przykładu wykorzystaj w następnym zadaniu. Po wpisaniu programu zmodyfikuj jego treść tak aby otrzymać trzy kwadraty w różnych miejscach o różnych wielkościach stykające się rogami np.



```

PROGRAM KWAD2;
USES
  CRT,GRAPH;
VAR
  STEROWNIK,TRYB:INTEGER;
PROCEDURE KWADRAT(X,Y,D:INTEGER);
BEGIN
  LINE(X,Y,X+D,Y);
  LINE(X+D,Y,X+D,Y+D);
  LINE(X+D,Y+D,X,Y+D);
  LINE(X,Y+D,X,Y);
END;
BEGIN
  STEROWNIK:=DETECT;{AUTOMATYCZNE WYKRYCIE RODZAJU KARTY}
                      {GRAFICZNEJ PRZEZ KOMPUTER}
  INITGRAPH(STEROWNIK,TRYB,'C:\UCZEN\TP\BGI');

```

{ W APOSTROFACH PODAJEMY GDZIE MAMY NAGRANE }
 { STEROWNIKI GRAFICZNE DLA KARTY GRAFICZNEJ }

```
KWADRAT(20,20,50);
REPEAT UNTIL KEYPRESSED;
END.
```

ZADANIE 52.

Wykorzystując instrukcję graficzną LINE(X1,Y1,X2,Y2) z czterema parametrami X1,Y1 – współrzędne początku odcinka oraz X2,Y2 współrzędne końca odcinka zdefiniować procedurę TROJKAT(X,Y,D) X,Y – współrzędne górnego wierzchołka trójkąta D–długość boku trójkąta. Wykorzystaj treść zadania poprzedniego w tym zadaniu. Wywołaj trzy razy procedurę TROJKAT tak aby otrzymać trzy różne trójkąty w różnych miejscach ekranu.

UWAGA: Procedura LINE przyjmuje tylko jako parametry wartości INTEGER. Celu zamiany typy REAL na INTEGER użyj funkcji ROUND np. ROUND(D/3)

ZADANIE 53. (na ocenę bardzo dobrą)

Zdefiniować procedurę ROMB(X,Y,D,ALFA)

X,Y – współrzędne dowolnego wierzchołka rombu D– długość boku rombu. Wywołaj trzy razy procedurę ROMB tak aby otrzymać trzy różne romby w różnych miejscach ekranu.

Program demonstracji grafiki

```
program demonstracja_grafiki;
uses      graph, {dołączenie grafiki}
          crt; {dołączenie modułu ekranu }
var
  sterownik, tryb : integer;
  x,y,i : integer;
  dalej:char;
begin
  sterownik:=detect;
  {CGA          320x200 1 strona}
  {Hercules     720x348 2 strony}
  {VGA          640x480 1 strona}
  initgraph(sterownik,tryb,'a:\tp'); {ZAINICJOWANIE GRAFIKI}
                                     {a:\tp – określenie gdzie są sterowniki graficzne BGI}
  { ***** }
  { współrzędne ekranu w trybie graficznym }
  (0,0)                                     (getmaxx,0)

  (0,getmaxy)                             (getmaxx,getmaxy)

  { ***** }
  setlinestyle(3,3,1); {Pierwsza zmienna określa rodzaj linii:}
                        {0–linia ciągła }
                        {1–linia kropkowa}
                        {2–linia centrowana}
                        {3–linia przerywana}
                        {4–linia zdefiniowana}
                        {druga zmienna określa sposób rysowania linii}
                        {może mieć wartość 0–11}
                        {trzecia zmienna określa grubość linii:}
                        {1–linia cienka}
                        {3–linia pogrubiona}
  line(100,100,150,100); {instrukcja rysuje odcinek}
                        {pierwsze dwie zmienne określają współ. początku a }
                        {drugie dwie współrzędne końca odcinka}

  line(100,100,100,150);
  line(100,150,150,150);
  line(150,150,150,100);
  settextrstyle(0,0,0); {Pierwsza zmienna określa krój pisma:}
                        {0–standardowy}
                        {1–kreskowy potrójny}
                        {2–kreskowy indeksowy}
                        {3–kreskowy bezszeryfowy}
```

```

                                {4-kreskowy gotycki}
                                {Droga zmienna określa kierunek druku;}
                                {0-poziomo od lewej do prawej}
                                {1-pionowo od dołu do góry}
                                {Trzecia zmienna określa rozmiar czcionki}
                                {może mieć wartość 0–10}
outtextxy(80,125,'A'); {Instrukcja ta drukuje tekst na ekranie}
                                {pierwsze dwie zmienne określają pozycje tekstu}
                                {trzecia zmienna zawiera tekst do wydruku}

repeat until keypressed; {zatrzymanie wykonywania programu}
cleardevice;             {czyszczenie ekranu graficznego}

settextstyle(4,1,4);
moveto(50,50);           {przeniesienie kursora graficznego do punktu}
                                {o danych współrzędnych}

outtext('PASCAL');      {wyprowadzenie tekstu od miejsca ostatniego}
                                {położeni kursora graficznego}

readln;
repeat until keypressed; {zatrzymanie wykonywania programu}
cleardevice;             {czyszczenie ekranu graficznego}
putpixel(180,180,1);
settextstyle(0,0,0);
outtextxy(50,60,'teraz rysowanie punktu na srodku ekranu');
putpixel(getmaxx div 2,getmaxy div 2,2);
readln;
repeat until keypressed; {zatrzymanie wykonywania programu}
cleardevice;             {czyszczenie ekranu graficznego}
setfillstyle(2,3);
bar(20,20,100,100);     {wykreślenie słupka jednowymiarowego}
                                {pierwsza i druga zmienna współrzędne górnego lewego wierzchołka}
                                {trzeci i czwarta zmienna to współrzędne prawego dolnego rogu}

readln;
repeat until keypressed; {zatrzymanie wykonywania programu}
cleardevice;             {czyszczenie ekranu graficznego}
setfillstyle(2,3);
bar3d(110,110,180,180,20,topon);
                                {rysowanie wykresu słupkowego trójwymiarowe}
                                {pierwsza i druga zmienna współrzędne górnego lewego wierzchołka}
                                {trzeci i czwarta zmienna to współrzędne prawego dolnego rogu}
                                {piąta współrzędna gdy}
                                {topon – zaznaczenie górnej powierzchni}
                                {topoff – bez zaznaczenia górnej pow.}

readln;
repeat until keypressed; {zatrzymanie wykonywania programu}
cleardevice;             {czyszczenie ekranu graficznego}
arc(100,100,30,300,50); {rysowanie łuku}
                                {dwie pierwsze współrzędne to środek łuku}
                                {trzecia współrzędna to kąt rozpoczęcia w stopniach}
                                {czwarta współrzędna to kąt zakończenia w stopniach}
                                {piąta współrzędna to promień łuku}

readln;
repeat until keypressed; {zatrzymanie wykonywania programu}
cleardevice;             {czyszczenie ekranu graficznego}
circle(100,100,70);     {rysowanie okręgu}
                                {dwie pierwsze współrzędne są środkiem okręgu}
                                {trzecia współrzędną jest promieniem}

readln;
repeat until keypressed; {zatrzymanie wykonywania programu}
cleardevice;
ellipse(300,150,0,360,200,50); {Pierwsze dwie zmienne określają;}
                                {współrzędne środka elipsy}
                                {Drugie dwie zmienne określają;}
                                {kąt początkowy i końcowy elipsy}
                                {Piąta i szósta zmienna określa}
                                {średnicę poziomą i pionową elipsy}

setfillstyle(2,3);      {instrukcja definiującą wzorec wypełnienia}
                                {Pierwsza zmienna określa numer wzorca;}
                                {0-wypełnienie kolorem tła}
                                {1-wypełnianie zadany kolorem}
                                {2-wzor ———}
                                {3-wzor ////}

```

```

{4-pogrubiony wzór ///}
{5-wzor \\\}
{6-pogrubiony wzór \\\}
{7-krata prosta}
{8-krata skośna}
{9-krata skośna z przeplotem}
{10-wzor kropkowy}
{11-gesty wzór kropkowy}
{12-wzor zdefiniowany przez użytkownika}
{Druga zmienna określa numer koloru wzoru}
{0-czarny}
{1-niebieski}
{2-fioletowy}
{3-kolor tła (biały)}

floodfill(300,150,3);

{Procedura wypełniania obszaru zadany wzorem}
{Pierwsza i druga zmienna deklarują}
{początek wypełniania obszaru}
{trzecia zmienna określa kolor ograniczenia}

readln;
repeat until keypressed;
setviewport(130,130,250,250,true);

{zatrzymanie wykonywania programu}

{definiowanie okna graficznego}
{dwie pierwsze zmienne to górny lewy róg okna}
{dwie następne zmienne to dolny prawy róg okna}
{piątą zmienną gdy true- obcinanie rysunków do}
{rozmiarów okna gdy false-bez obcinania}

clearviewport;
readln;
repeat until keypressed;
setviewport(0,0,719,347,true);
cleardevice;
readln;
repeat until keypressed;
cleardevice;
pieslice(150,100,0,90,50);

{czyszczenie ekranu graficznego }

{zatrzymanie wykonywania programu}

{definiowanie okna graficznego}
{czyszczenie ekranu graficznego}

{Procedura rysująca wycinki kołowe}
{ pierwszy parametr- współrzędna X środka okręgu}
{ drugi współrzędna Y środka okręgu}
{trzeci początek wycinka w stopniach }
{ czwarty koniec wycinka w stopniach }
{ piąty promień koła}

readln;
repeat until keypressed;
cleardevice;
setactivepage(0);
circle(100,100,50);
setactivepage(1);
circle(100,100,20);
for i:=1 to 50 do
begin
    setvisualpage(round(i div 2));
    delay(100);
end;

{aktywizacja zerowej strony graficznej}
{rysowanie na zerowej stronie graficznej}
{aktywizacja pierwszej strony graficznej}
{rysowanie na pierwszej stronie graficznej}

{wywoływanie na przemian strony }
{zerowej i pierwszej }

readln;
repeat until keypressed;
closegraph;

{zamknięcie grafiki wejście do trybu tekstowego}
{gdy nie jest znana karta program może sam ustalić kartę}
{graficzna i sam zainicjować grafikę }

sterownik:=detect;
initgraph(sterownik,tryb,"");
circle(round(getmaxx/2),round(getmaxy/2),round(getmaxy/2));
{ustalenie karty graficznej}
{getmaxx i getmaxy maksymalne wartości x i y dla danej karty graficznej}

readln;
repeat until keypressed;
restorecrtmode;
gotoxy(10,30);
textattr:=(1);
writeln('to jest tryb znakowy');
delay(2000);
textattr:=(9);

{przejdź do trybu znakowego}
{umieszczenie kursora tekstowego}
{gdy textattr=1 to będzie wydruk podkreślony}
{gdy textattr=9 to wydruk podkreślony i zwiększona jaskrawość}

```



```
writeln('to jest tryb znakowy');
delay(2000);
textattr:=(112);           {gdzy textattr=112 to wydruk będzie w negatywie}
writeln('to jest tryb znakowy');
delay(2000);
setgraphmode(0);           {przywrócenie trybu graficznego}
outtextxy(100,100,'GRAFIKA');
readln;
repeat until keypressed;
closegraph;                 {zamkniecie grafiki wejście do trybu tekstowego}
end.
```

Instrukcja wyboru CASE

Instrukcja wyboru CASE ma postać

```
case S of
    L1:I1;
    L2:I2;
    .....
    .....
    .....
    Ln:In
end;
```

gdzie: S jest selektorem(wybieraczem)—zmienna lub wyrażenie
 L1,L2,...,Ln lista etykiet(wartości jakie może przyjmować selektor S)
 I1,I2,...,In są instrukcjami wyboru

Działanie instrukcji CASE polega na obliczeniu wartości selektora S i wykonaniu tej instrukcji I1,I2,...,In, której wartość L1,L2,...,Ln odpowiada wartości selektora S.

Przykład

W instrukcji CASE można wpisywać również przedziały oraz instrukcje złożone. Oto przykład takiej instrukcji(fragment programu).

```
CASE I OF
    -10..-1 : BEGIN
        D:=SQR(I);
        WRITELN('DLA I=',I:3,' D=',D:4);
    END;
    1..MAX: BEGIN
        D:=SQR(I)*I
        WRITELN('DLA I=',I:2,' D=',D:4);
    END
END;
```

PRZYKŁAD 24

Program, który po podaniu wartości kąta powie do której ćwiartki on należy.

```
PROGRAM CWIARTKI;
USES
    CRT;
VAR
    KAT:INTEGER;{TA ZMIENNA MUSI BYĆ INTEGER}
BEGIN
    WRITELN('PODAJ KAT=');
    READLN(KAT);
    CASE KAT DIV 90 OF
        0:WRITELN('PIERWSZA CWIARTKA');
        1:WRITELN('DRUGA CWIARTKA');
        2:WRITELN('TRZECIA CWIARTKA');
        3:WRITELN('CZWARTA CWIARTKA')
    END;
    REPEAT UNTIL KEYPRESSED:
END.
```

ZADANIE 54.

Napisać program podający ile wynosi reszta z dzielenia liczby przez 5 (liczba wczytana z klawiatury). Użyj instrukcję CASE oraz MOD. Nazwa programu z53_cztery_pierwsze_litery_nazwiska.pas

ZADANIE 55.

Napisać test komputerowy składający się z czterech pytań. Każde pytanie powinno być procedurą. Do oceny poprawności odpowiedzi na każde pytanie zastosuj instrukcję CASE. Program powinien zapamiętywać liczbę poprawnych odpowiedzi oraz informować czy odpowiedź była poprawna czy niewłaściwa. Stopień końcowy wystawiany powinien być z zastosowaniem instrukcji CASE.

ilość poprawnych odpowiedzi	0	1	2	3	4
Skala ocen	Ndst.	Dop.	Dst	Db	bdb

Instrukcja losowa

W wielu przypadkach przy pisaniu programów konieczne jest stosowanie liczb losowych. Liczby losowe jest to ciąg przypadkowo dobranych liczb z określonego przedziału.

W programowaniu w języku PASCAL stosowane są dwie instrukcje do generowania liczb losowych. RANDOMIZE– zainicjowanie generatora liczb losowych.

RANDOM(W)–wygenerowanie(utworzenie) liczby losowej.

W– liczba integer jest górna wartość przedziału z jakiego zostaną wylosowane liczby losowe czyli będzie to liczby z przedziału $<0,W)$ W nie należy do przedziału. Należy przed pierwszym wywołaniu instrukcji RANDOM umieścić instrukcję RANDOMIZE. Gdy nie zastosujemy instrukcji RANDOMIZE to programu będzie podawał ten sam ciąg liczb losowych przy każdym uruchomieniu programu.

PRZYKŁAD 25

Program generujący liczby losowe z przedziału od $<0,10)$

```
PROGRAM LOSOWA;
USES
  CRT;
VAR
  A:CHAR;
BEGIN
  CLRSCR;
  RANDOMIZE;
  REPEAT
    WRITELN(RANDOM(10));
    A:=READKEY; {instrukcja READKEY–wprowadzenie znaku z}
                {klawiatyry bez wyprowadzenia na monitor}
  UNTIL (A='N')OR(A='n');
END.
```

ZADANIE 56.

Uwaga treść poprzedniego przykładu wykorzystaj do tego zadania.

Napisać program generujący liczby z przedziału $<-7,4>$.

PRZYKŁAD 26

Przy pomocy komputera można zasymulować ruchy Browna. Ruchy Browna są przypadkowymi ruchami cząsteczki powietrza. Program pokazuje ruch tej cząsteczki. Zmiana toru tej cząsteczki jest wynikiem zderzenia z innymi cząsteczkami.

```
PROGRAM RUCHY_BROWNA;
USES
  CRT,GRAPH;
VAR
  STEROWNIK,TRYB:INTEGER;
  XSTART,YSTART:INTEGER;
  D_X,D_Y:INTEGER;
{D_X,D_Y PRZYROST WARTOSCI WSPOLRZEDNYCH EKRANU}
BEGIN
  RANDOMIZE;
```

```

STEROWNIK:=DETECT;
{DETECT ZIDENTYFIKOWANIE KARTY GRAFICZNEJ PRZEZ KOMPUTER}
  INITGRAPH(STEROWNIK,TRYB,"");
  XSTART:=(GETMAXX DIV 2);
  YSTART:=(GETMAXY DIV 2);
{GETMAXX,GETMAXY –OKRESLENIE MAKSYMALNYCH WARTOSCI EKRANU}
  MOVETO(XSTART,YSTART);
{MOVETO PERZENIESIENIE KURSORA GRAFICZNEGO DO XSTAR,YSTART}
  REPEAT
    D_X:=-7+RANDOM(16);
    D_Y:=-5+RANDOM(10);
    LINEREL(D_X,D_Y);
{LINEREL WYKRERSLЕНИЕ LINI PROSTEJ OD KURSORA GRAFICZNEGO O
PRZYROST D_X I D_Y}
    DELAY(200);
{DELAY(CZAS_ZWLOKI) CZAS_ZWLOKI=1000 TO SEKUNDA}
  UNTIL KEYPRESSED;
  CLOSEGRAPH;
END.

```

ZADANIE 57.

W celu przetestowania poprawności generatora liczb losowych można napisać następujący program: losowo zapalając punkty na ekranie. Jeżeli rozkład punktów na ekranie jest równomierny to oznacza, że generator działa poprawnie. Przy generowaniu użyj instrukcji GETMAXX oraz GETMAXY. Po wykonaniu tej części zadania możesz zaprogramować losowy kolor zapalanego punktu.

ZADANIE 58. (na ocenę bardzo dobrą)

Napisz program losujący n różnych liczb ze zbioru $\{1 \dots m\}$ możliwych $n \leq m$, m,n wczytywane z klawiatury.

ZADANIA PRZYKŁADOWE NA SPRAWDZIAN (CZAS 85 MINUT)

ZADANIE1

Narysować w trybie graficznym trapez równoramienny oraz oznaczyć jego wierzchołki literami.

ZADANIE2

Napisać program rysujący losowo wybrane okręgi na monitorze. Losowo wybierany jest punkt środka okręgu oraz promień. Napisz tak program aby okręgi nie mieszczące się na ekranie nie były rysowane. Rysowanie następnego okręgu po wciśnięci klawisza ENTER.

ZADANIE3

Narysować w trybie tekstowym postać człowieka przy wykorzystaniu procedur:

–GLOWA

–RECE

–TULW

–NOGI

wywołaj procedury tak aby powstała postać człowieka

ZADANIE4

Ztablicować funkcję z krokiem 0.3 przy wykorzystaniu definiowania funkcji

$$f(x) = \begin{cases} x^2 - 12x + 36 & \text{dla } < 5,6 > \\ 1 & \text{dla } < \frac{\pi}{4}, 5) \\ tg(x) & \text{dla } < 0, \frac{\pi}{4} \end{cases}$$

ZADANIE5

Narysować w trybie graficznym prostokąt. Zdefiniuj procedurę PROST z czterema parametrami X,Y – współrzędne lewego górnego rogu prostokąta

A – szerokość prostokąta

B – długość prostokąta

Wywołaj procedurę PROST z takimi parametrami aby narysować na ekranie prostokąt

ZADANIE6

Napisać program obliczający pole powierzchni oraz obwód prostokąta. Zdefiniuj funkcje POW oraz OBW .Wywołaj funkcje POW oraz OBW z A=4 i B=5. Wypisz na ekranie wyniki w odpowiednim formacie oraz z jednostkami. Napisz tak program ,aby komputer pytał się o A i B następnie liczył POW i OBW dla różnych A i B. Wypisz na ekranie wyniki w odpowiednim formacie oraz z jednostkami.

ZADANIE 7

Napisz program z użyciem funkcji CASE, który po podaniu pierwszej litery kraju, z którym graniczy Polska odpowie, jaki to kraj. Np. po podaniu „c” lub „C” wyświetlone zostanie „Czechy”. Wielkość liter nie będzie miała znaczenia. Gdy litera nie będzie prawidłowa odpowie „nie ma takiego kraju”.

ZADANIE 8

Napisać program symulujący rzut kostką do gry. Podczas symulacji widoczna jest tylko jedna ściana kostki. Widok ściany zapisany jest w procedurze (czyli będzie sześć procedur). Wybór procedury za pomocą instrukcji CASE. Następne losowanie przy użyciu klawisza ENTER koniec klawisz K.

Ilość zadań	0–2	3	4	5	6	7-8
Ocena	Niedostateczny	Dopuszczający	Dostateczny	Dobry	Bardzo dobry	Celujący

CZĘŚĆ SZÓSTA WYKŁADU Z PASCALA

Funkcje tekstowe, rekordy.

OPERACJE I FUNKCJE WYKONYWANE NA TEKSTACH

S1+S2	sklejenie (łączenie, dodawanie) łańcuchów S1 i S2
CONCAT(S1,S2)	sklejenie łańcuchów S1 i S2
LENGTH(S)	zwraca długość łańcucha S
COPY(S,N1,N2)	zwraca łańcuch N2 znaków wycięty z łańcucha S poczynając od pozycji N1
POS(CH,S)	zwraca pozycję pierwszego wystąpienia znaku CH w łańcuchu S
INSERT(S1,S2,k)	wstawia łańcuch S1 do łańcucha S2 poczynając od pozycji k
DELETE(S,N1,N2)	usuwa N2 znaków z łańcucha S poczynając od pozycji N1
STR(R,S)	zamiana wartości R na napis S
VAL(S,R,k)	zamiana napisu S na wartość R gdy k=0 to konwersja nastąpiła prawidłowo gdy k>0 to wartość k mówi, który znak w S nie poddał się zamianie
S[n] := CH	ustawia na pozycji n w łańcuchu S znak CH
S[0] := CH	zmienia długość dynamiczną łańcucha S na ORD(CH)

Operacje porównywania tekstów

Operatory < , > , = , <>

Porównywanie odbywa się na zasadzie przeszukiwania łańcucha do pierwszej niezgodności i porównywanie kodów nie zgadzających się znaków np.

'Fortran' < 'Pascal' bo ('F' < 'P')

'fortran' > 'Pascal' bo ('f' > 'P')

PRZYKŁAD 27

Program, który wczytuje tekst następnie drukuje ten tekst litera po literze w pionie

Polecenia do wykonania:

- Zanotuj w zeszyte przedmiotowym a następnie naucz się informacje dotyczące:
 - operacje i funkcje wykonywane na tekstach
 - operacje porównywania tekstów

specyfikacja problemu

dane wejściowe:

łańcuch tekstowy wczytywany z klawiatury

dane wyjściowe:

tekst wypisywany litera po literze na ekranie w pionie

Zliczenie przykładu:

- sprawdzenie notatek,
- odpowiedź ustana,
- uruchomienie oraz wykonanie programu.

```
PROGRAM STRING1;
```

```
  USES
```

```
    CRT;
```

```
  VAR
```

```
    NAPIS1:STRING;
```

```
    I:INTEGER;
```

```
BEGIN
```

```
  CLRSCR;
```

```
  WRITELN('PODAJ TEKST');
```

```
  READ(NAPIS1);
```

```
  FOR I:=1 TO LENGTH(NAPIS1) DO
```

```
    WRITELN(NAPIS1[I]);
```

```
  REPEAT UNTIL KEYPRESSED;
```

```
END.
```

ZADANIE 59.

Napisz program, który po wczytaniu tekstu do zmiennej drukował będzie ten tekst na ekranie litera po literze w poziomie. Po wyprowadzeniu każdej litery nastąpi sygnał dźwiękowy. Wyprowadzanie liter następuje po pewnym czasie wczytywanym do programu jako dana.

Zliczenie zadania:

- sprawdzenie specyfikacji problemu zapisanej w zeszycie,
- uruchomienie oraz wykonanie programu.

ZADANIE 60.

Napisz program, który po wczytaniu tekstu wydrukuje go po przekątnej litera po literze zaczynając od miejsca x,y na ekranie. x,y- wczytywane do programu z klawiatury.

```
(x,y) np. (10,10)
  K
   A
    W
     A
```

ZADANIE 61.

Napisz program, który po wczytaniu tekstu oraz x, y wyświetli napisy w następujący sposób jak poniżej. Przykład dla wyrazu KWADRAT.

```
(x,y)
      KWADRAT
      W      A
      A      R
      D      D
      R      A
      A      W
      TARDAWK
```

ZADANIE 62.

Napisz program, który będzie z po wczytaniu tekstu oraz x, y wyświetli napisy w następujący sposób jak poniżej:

```
(x,y) np. (10,10)
      K  A  R  T
      W  D  A
```

ZADANIE 63.

Napisz program, który będzie zliczał ilość liter w tekście. Do programu wczytujemy się jako dane: tekst jako zmienną string oraz literę, której chcemy zliczyć częstotliwość występowania.

Zliczenie zadania:

- sprawdzenie specyfikacji problemu zapisanej w zeszycie,
- uruchomienie oraz wykonanie programu.

PRZYKŁAD 28

Program, który wczytuje się dwa teksty: wymiar oraz jednostkę, obliczy pole prostokąta.

Np. Po wczytaniu 4 cm oraz 5 cm wynikiem działania programu będzie:

Pole=20 cm²

```
Program demo_val;
Uses
    crt;
Var
    a_str,b_str:string[4];
    pole,bok_a,bok_b:real;
    k:integer;
begin
    clrscr;
    repeat
        write('podaj dlugosc boku <=9 lacznie z jednostka np. 5 cm a=');
        readln(a_str);
```

```

val(a_str[1],bok_a,k);

    if k<>0 then writeln('pierwszy znak powinien byc cyfra');
until k=0;

repeat
    write('podaj dlugosc boku <=9 lacznie z jednostka np. 6 cm b=');
    readln(b_str);
    val(b_str[1],bok_b,k);
    if k<>0 then writeln('pierwszy znak powinien byc cyfra');
until k=0;

pole:=bok_a*bok_b;
writeln('pole=',pole:6:2,' cm^2');
repeat until keypressed;
end.

```

ZADANIE 64.

Napisz program, który po wczytaniu dwóch liczb w systemie dwójkowym max 3 bity np. 101. Zamieni te liczby na system dziesiętny i wyświetli ich wartość. Następnie dodaj je i wyświetli sumę w postaci dwójkowej.

Wskazówka:

Do trzech zmiennych wczytaj wartości trzech bitów (0 lub 1). Następnie na tej podstawie oblicz wartość liczby w systemie dziesiętny, korzystaj ze wzoru:

$L = \text{liczba_systemu_dziesietnego}$

$L = (\text{wartosc_pierwszego_bit_od_lewej}) * 4 + (\text{wartosc_srodkowego_bitu}) * 2 + (\text{wartosc_trzeciego_bit_od_prawej}) * 1$

Zamiana liczby z systemu dziesiętnego na dwójkowy możesz dokonać poprzez rozpatrzenie wszystkich możliwości instrukcją if

np. 1001 to liczba 9.

ZADANIE 65.

Napisz program, który po wczytaniu liczby obliczy sumę jej cyfr. Np. 3468 suma=21

Wskazówka:

Liczbę wczytaj do zmiennej tekstowej, zamieniaj kolejne cyfry liczby na liczby i sumuj.

ZADANIE 66.

Napisz program, który po wczytaniu tekstu (WYRAZU) odpowie czy ten wyraz jest Palindromem.

Wyraz kajak jest Palindromem. Działanie programu:

TAK wyraz „kajak” jest Palindromem

NIE wyraz „owca” nie jest Palindromem

PRZYKŁAD 28a

Program do znajdowania na, którym miejscu w wczytanym łańcuchu znaków znajduje się wczytany podciąg.

```

PROGRAM STRING2;
  USES
    CRT;
  VAR
    TEKST:STRING;
    SZUKANY:STRING;
    MIEJSCE:INTEGER;
  BEGIN
    CLRSCR;
    WRITELN('PODAJ TEKST');
    READLN(TEKST);
    WRITELN('PODAJ SZUKANY TEKST');
    READLN(SZUKANY);
    MIEJSCE:=POS(SZUKANY,TEKST);
    WRITELN('MIEJSCE=',MIEJSCE);
    REPEAT UNTIL KEYPRESSED;
  END.

```

Zliczenie zadania:

- sprawdzenie specyfikacji problemu zapisanej w zeszycie,

- uruchomienie oraz wykonanie programu.

ZADANIE 67.

Napisz program, który po wczytaniu tekstu (WYRAZU → każda litera inna, wielkość liter nie ma znaczenia) i dwóch liter odpowie ile liter znajduje się pomiędzy tymi literami np.

Lato i litery „L” „o” wynik programu → 2

krowa i litery „k” „r” wynik programu → 0

PRZYKŁAD 29

Program, który będzie wycinał tekst z pomiędzy wczytanych liter

```
PROGRAM STRING2;
USES
  CRT;
VAR
  TEKST:STRING;
  LITERA1,LITERA2:CHAR;
  MIEJSCE1, MIEJSCE2, ILEZNAKOW:BYTE;
BEGIN
  CLRSCR;
  WRITELN('PODAJ TEKST');
  READLN(TEKST);
  WRITELN('PODAJ PIERWSZA LITERĘ');
  READLN(LITERA1);
  WRITELN('PODAJ DRUGA LITERĘ');
  READLN(LITERA2);
  MIEJSCE1:=POS(LITERA1,TEKST);
  MIEJSCE2:=POS(LITERA2,TEKST);
  ILEZNAKOW:=ABS(MIEJSCE1-MIEJSCE2);
  WRITELN('STARY=',TEKST);
  IF MIEJSCE1<MIEJSCE2 THEN
    DELETE(TEKST,MIEJSCE1+1,ILEZNAKOW-1)
  ELSE
    DELETE(TEKST,MIEJSCE2+1,ILEZNAKOW-1);
  WRITELN('NOWY=',TEKST);
  REPEAT UNTIL KEYPRESSED;
END.
```

Zliczenie zadania:

- sprawdzenie specyfikacji problemu zapisanej w zeszycie,
- uruchomienie oraz wykonanie programu.

ZADANIE 68.

Napisz program, który po wczytaniu tekstu (WYRAZ) i dwóch liter,

pierwsza litera → jaką literę zastępujemy

druga litera → na jaką literę zamienić

np. wyraz → mama

pierwsza litera „m”

druga litera „t”

wynik programu → tata

ZADANIE 69.

Napisz program, który będzie zliczał ilość znaków w tekście. Wygląd działania programu dla tekstu matematyka:

a - 3 razy

m - 2 razy

t - 2 razy

y - 1 raz

e - 1 raz

k - 1 raz

ZADANIE 70.

a) zapisz w zeszycie następującą kolumnę liczb:

$2^0=1$

$2^1=2$

$2^2=4$

.....

.....

 $2^{10}=?$

b)zamień liczbę z systemu dziesiętnego 220+numer_w_dzienniku korzystając z kolumny liczb z podpunktu a) na system dwójkowy.

c)zamień liczbę z systemu dziesiętnego 220+numer_w_dzienniku korzystając z dzielenia z resztą na system dwójkowy(patrz poniżej).

Aby z liczby dziesiętnej uzyskać odpowiadającą jej liczbę dwójkową należy dzielić daną liczbę przez 2, wyniki kolejnych dzielen zapisujemy w słupku reszty z dzielen zapisujemy po prawej stronie za kreską, kolejne dzielenia wykonujemy do momentu aż uzyskamy wynik z dzielenia mniejszy niż 1.
Teraz wystarczy przepisać uzyskane reszty z dzielen od dołu do góry: 10101101

$(173)_{10}=(10101101)_2$

173 : 2	reszta	1
86 : 2	reszta	0
43 : 2	reszta	1
21 : 2	reszta	1
10 : 2	reszta	0
5 : 2	reszta	1
2 : 2	reszta	0
1 : 2	reszta	1

d)Napisz program, który zamieni liczbę dziesiętną na liczbę w systemie dwójkowym. Użyj definiowania funkcji:
function Dec_2_bin(N:integer):string; Program zapyta się o liczbę naturalna i zamieni ją na system dwójkowy
wygląd ekranu np.

Podaj liczbę naturalna=209

$(209)_{10}=(11010001)_2$

REKORDY

REKORD – jest strukturą składającą się ze stałej liczby składników, nazywanych polami. Pola mogą być różnych typów, każde z nich ma nazwę–identyfikator, który używany jest do jego wybierania.

NAZWISKO	IMIĘ	WZROST	WAGA	} R E K O R D
----------	------	--------	------	------------------------------

NAZWISKO IMIĘ WZROST WAGA to są pola o takich nazwach (identyfikatorach)

PRZYKŁAD 30

Wpisać dane osobowe o pracownikach do struktury danych w postaci rekordu.

PROGRAM DANE;

Uses crt;

TYPE {POCZĄTEK DEFINIOWANIA REKORDU}

DANEOSOBOWE=RECORD

NAZWISKO: STRING[20];

IMIĘ : STRING[15];

WIEK : INTEGER;

WAGA : REAL

{NAZWA POLA TYP POLA}

END; {KONIEC DEFINIOWANIA REKORDU}

VAR

OSOBA1,OSOBA2:DANEOSOBOWE;

{ZADEKLAROWANIE DANYCH O DWÓCH OSOBACH JAKO REKORDY}

BEGIN

WRITELN('PODAJ DANE O PIERWSZEJ OSOBIE');

```

READLN(OSOBA1.NAZWISKO);
READLN(OSOBA1.IMIE);
READLN(OSOBA1.WIEK);
READLN(OSOBA1.WAGA);
WRITELN('PODAJ DANE O DRUGIEJ OSOBIE');
READLN(OSOBA2.NAZWISKO);
READLN(OSOBA2.IMIE);
READLN(OSOBA2.WIEK);
READLN(OSOBA2.WAGA);
WRITELN('POSIADAMY INFORMACJE O NASTĘPUJĄCYCH OSOBACH');
WRITELN('    OSOBA PIERWSZA');
WRITELN('NAZWISKO  ',OSOBA1.NAZWISKO:20);
WRITELN('IMIE      ',OSOBA1.IMIE:20);
WRITELN('WIEK      ',OSOBA1.WIEK:20);
WRITELN('WAGA      ',OSOBA1.WAGA:20:2);
WRITELN('    OSOBA DRUGA');
WRITELN('NAZWISKO  ',OSOBA2.NAZWISKO:20);
WRITELN('IMIE      ',OSOBA2.IMIE:20);
WRITELN('WIEK      ',OSOBA2.WIEK:20);
WRITELN('WAGA      ',OSOBA2.WAGA:20:2);
Repeat until keypressed;
END.

```

Uwaga1:

W celu dokonania operacji na danych zapisanych w postaci rekordu (wczytanie wypisanie) należy podać nazwę rekordu oraz po kropce nazwę pola np. osoba1.waga

osoba1 → nazwa rekordu

waga → nazwa pola

Uwaga2:

W zadaniu został wykorzystany typ napisowy STRING w nawiasie kwadratowym piszemy ile maksymalnie może mieć liter zmienna typu STRING w używanym przez nas programie. Zmienna ta może mieć maksymalnie 255 znaków.

ZADANIE 71.

Napisz program do tworzenia bazy danych. Baza ta będzie przechowywać nazwę produktu oraz jego cenę. Program wczytuje cztery rekordy oraz wyświetla ich zawartość w postaci tabeli (bez ramek → czyli równe kolumny, niezależnie od wielkości/długości danych). Nazwy pól rekordów, z trzema literami nazwiska ucznia. Np. waga_kow

ZADANIE 72.

Napisać (program obliczający długość wektora w przestrzeni. Zadeklarować (record o nazwie PUNKT, który będzie miał trzy pola X – pierwsza współrzędna punktu Y,Z – następne współrzędne punktu. Następnie zadeklarować dwa punkty w przestrzeni o nazwach W1 W2. Następnie obliczyć długość wektora operując na polach recordów.

ZADANIE 73.

Napisać program na dodawanie, odejmowanie, mnożenie dzielenie liczb zespolonych. Działania te powinny być procedurami. Procedurą powinno być również wczytywanie i wyprowadzanie danych. Procedura wyprowadzania danych powinna być tak wykonana, że są zapisane liczby w postaci $ZW=a+j*b$. Program powinien prezentować możliwe działania do wykonania. Wybór działania zadanego przez użytkownika powinien odbywać się za pomocą instrukcji CASE. Zapisz dane liczby zespolone jako Z1 i Z2, wynik jako ZW. Z1,Z2,ZW zdefiniowane jako record o polach R,U.

R – część rzeczywista U – część urojona.

Podstawy teoretyczne.

Liczba zespolona może być przedstawiona w postaciach:

- algebraicznej
- trygonometrycznej
- wykładniczej

W zadaniu korzystać będziemy z postaci algebraicznej

Gdy mamy dwie liczby zespolone

$Z1=a1+j*b1$ np. $Z1=2+3*j$

oraz

$Z2=a2+j*b2$ np. $Z2=4+j*5$ to

$a1=2$ $b1=3$ $a2=4$ $b2=5$.

gdzie:

$$j = \sqrt{-1} \quad \text{czyli} \quad j*j = -1$$

dodawanie:

$$Z1+Z2=(a1+j*b1) + (a2+j*b2)=a1+a2 +j*(b1+b2) = aw + j*bw$$

$$\text{np. } Z1+Z2=(2+3*j)+(4+5*j)=2+4+(3+5)*j=6+j*8$$

odejmowanie:

$$Z1-Z2=(a1+j*b1) - (a2+j*b2)=a1-a2 +j*(b1-b2) = aw + j*bw$$

$$\text{np. } Z1-Z2=(2+3*j)-(4+5*j)=2-4+(3-5)*j=-2-j*2$$

mnożenie

$$Z1*Z2=(a1+j*b1) * (a2+j*b2)=a1*a2-b1*b2 + j(b2*a1+b1*a2) \text{ np.}$$

$$Z1*Z2=(2+3*j)*(4+5*j)=2*4-3*5 +j(3*4+2*5)=-7+j*22$$

dzielenie

$$Z1/Z2=(a1+j*b1)/(a2+j*b2)=(a1*a2+b1*b2)/(a2*a2+b2*b2)+$$

$$+j*(b1*a2-b2*a1)/(a2*a2+b2*b2)$$

Gdy $Z2=0$ czyli $a2=0$ oraz $b2=0$ to działanie $Z1/Z2$ jest niewykonalne. Program powinien uwzględnić ten przypadek.

ZADANIA PRZYKŁADOWE NA SPRAWDZIAN (CZAS 85 MINUT)

ZADANIE 1

Oblicz cenę gazety komputerowej w systemie dziesiętnym w Kraju Bajtlandia. W kraju tym ceny zapisywane są w systemie dwójkowym.

Cena wpisywana jest następująco, do zmiennej tekstowej:

Jeden_bajt_spacja_nazwa_gazety

np. 11111111 Komputer

cenę w Bajtlandi zapisz do zmiennej String.

Dla takiej danej na ekranie zobaczymy:

cena=255

ZADANIE 2

Napisz program, który po wczytaniu tekstu oraz x, y wyświetli napisy w następujący sposób jak poniżej. Przykład dla wyrazu KWADRAT.

(x,y)	→	np. (10,10)	Użyj trzech pęli for.
T		T	
AA		A	
R R		R	
D D		D	
A A		A	
W		WW	
K		K	

ZADANIE 3

Za pomocą RECORDÓW zapisać dane o czterech uczniach. Dane obejmują nazwisko imię oraz oceny z czterech przedmiotów. Wydrukować dane w postaci zestawienia (bez ramki).

	PRZEDMIOT1	PRZEDMIOT2	PRZEDMIOT3	PRZEDMIOT4
UCZEN1	?	?	?	?
UCZEN2	?	?	?	?
UCZEN3	?	?	?	?
UCZEN4	?	?	?	?

Wyrównywanie danych w tabelach wykonaj bez użycia spacji.

ZADANIE 4

Dwa punkty $A(x_1, y_1)$ $B(x_2, y_2)$, które tworzą prostą zapisać w postaci recordu. Za pomocą instrukcji CASE dokonać wyboru opcji realizowanej przez program. Każda opcja stanowi osobną procedurę. Napisana jest również procedura wczytywania danych. Opcje programu są następujące:

–podanie wzoru funkcji w postaci $y=a*x+b$

–określenie czy funkcja jest rosnącą malejącą czy stała.

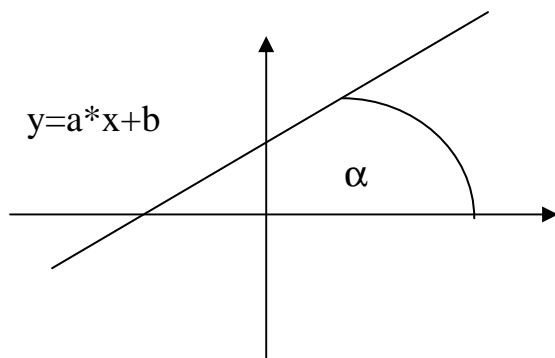
–określenie przedziałów gdzie funkcja jest dodatnia gdzie ujemna

–obliczanie wartości funkcji dla wczytanego argumentu z klawiatury.

Oblicz wzór funkcji liniowej w postaci kierunkowej, czyli $y = a*x + b$ oraz podaj kąt, jaki tworzy ta prosta z osią X.

Wskazówka:

Po podaniu dwóch różnych punktów $A(x_a, y_a)$ oraz $B(x_b, y_b)$ przez które przechodzi ta prosta.



Rozwiązanie:

1) Oblicz współczynnik kierunkowy a ze wzoru

$$a = \frac{yB - yA}{xB - xA}$$

2) Oblicz współczynnik **b** ze wzoru

$$b = yA - a * xA$$

3) Oblicz kąt alfa w radianach ze wzoru

$$alfa_x = \arctan(a)$$

4) Oblicz kąt w stopniach ze wzoru

$$alfa_s = \frac{alfa_x * 180}{\Pi}$$

Wprowadź A(0,1) B(2,2) a otrzymasz

a = 1 b = 1 alfa_s = 45 stopni

ZADANIE 5

Napisać program wykonujący następujące działania na liczbach zespolonych.

➤ $ZW = 2 * Z2 - Z3 + Z1$

Działanie powinny być procedurą. Procedurą powinno być również wczytywanie danych oraz wypisanie danych.

Zapisz dane liczby zespolone jako Z1 Z2 Z3 ,wynik jako ZW zdefiniowane jako typ programisty o nazwie zespolona_dana.

ZADANIE 6

Napisz program, który po wczytaniu tekstu (bez polskich liter) zamieni pierwszą literę małą na duże i między litery wpisze spacje..

np. mama zamieni na M_a_m_a (_ oznacza spację)

Ilość zadań	0-1	2	3	4	5	6
Ocena	Niedostateczny	Dopuszczający	Dostateczny	Dobry	Bardzo dobry	celujący

CZĘŚĆ SIÓDMA WYKŁADU Z PASCALA MODUŁY, GRAFIKA ŻÓŁWIA, REKURENCJA, FRAKTALE MODUŁY

Definicja:

Moduł jest zbiorem skompilowanych procedur, które mogą być dołączone do programu i wykorzystywane w nim.

Dołączanie modułów

Zachowując strukturę programu w języku PASCAL należy napisać:

uses lista_nazw_modułów;

Moduły standardowe

System (wszystkie procedury standardowe)
 Crt (obsługa ekranu, klawiatury, głośnika)
 Dos (wywołanie systemowe MS-DOS)
 Graph (procedury grafiki wysokiej rozdzielczości)
 Turbo3 (procedury kompatybilne z Turbo Pascalem 3.0)
 Graph3 (grafika żółwia)
 printer (obsługa drukarki)
 Overlay (obsługa nakładek)

Struktura modułu

unit nazwa_modułu; {ta sama nazwa musi być na dysku}

interface

uses nazwy_wykorzystywanych_modułów;

definicja stałych publicznych;

definicja typów publicznych;

deklaracja zmiennych publicznych;

deklaracja procedur publicznych;

implementation

definicja stałych prywatnych;

definicja typów prywatnych;

deklaracja zmiennych prywatnych;

deklaracja procedur prywatnych;

definicje procedur publicznych;

begin

instrukcje inicjujące moduł

end.

opis:

1) zamiast słowa program stosujemy słowo unit

2) moduł składa się z trzech sekcji

interface

publiczne oznacza, że zadeklarowane tutaj elementy są widziane przez program wywołujący moduł,

tutaj są tylko deklarowane a nie definiowane

implementation

prywatne oznacza, że zadeklarowane tutaj elementy są widziane tylko przez moduł, tutaj są również

definiowane zadeklarowane w interface funkcje i procedury

sekcja inicjująca moduł od słowa begin do end.

PRZYKŁAD 31

Unit modu1;

interface

const

g=9.81;

function Droga(czas:real):real;

implementation

function Droga(czas:real):real;

begin

droga:=g*czas*czas/2;

end;

begin

Writeln('to jest inicjacja modułu');

end.

Kompilowanie modułów

Moduły kompiluje się tak samo jak programy lecz po kompilacji pojawia się zbiór z rozszerzeniem TPU (wtedy możesz użyć modułu w innym programie). Kompilujemy na dysk.

Grafika żółwia

Pod tym pojęciem kryje się specjalny sposób rysowania w grafice wysokiej rozdzielczości. Dysponujesz żółwiem, który porusza się po ekranie (oczywiście zgodnie z twoim życzeniem) swoim ogonem zostawia ślad.

Żółw umie wykonywać następujące czynności:

NP(ilość_kroków)

ruch żółwia o określoną ilość kroków do przodu. Do przesuwu żółwia użyj instrukcji LINEREL(dX,dY) rysowanie linii od ostatniego miejsca kursora graficznego o wzrost dX oraz dY dX oraz dY oblicz z funkcji trygonometrycznych znając o ile ma przesunąć się żółw N oraz aktualny KAT.

BK(ilość_kroków)

ruch żółwia o określoną ilość kroków wstecz

LEWO(kat)

obrót nosa żółwia o określony kąt w lewo

PRAWO(kat)

obrót nosa żółwia o określony kąt w prawo

INICJUI_ZOLWIA

po tej instrukcji żółw ustawi się nosem na wschód i na środku ekranu

ZADANIE 74.

Napisz moduł ZOLW, w który będą zdefiniowane wszystkie procedury:

NP(ilość_kroków)

BK(ilość_kroków)

LEWO(kat)

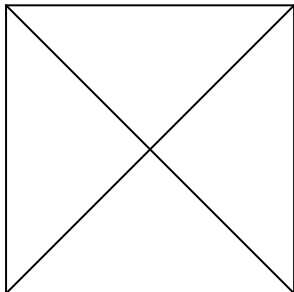
PRAWO(kat)

INICJUI_ZOLWIA

Moduł będzie miał zmienną globalną KAT, której wartość będzie zawierała aktualny kąt położenia nosa żółwia. Dokonaj kompilacji.

ZADANIE 75.

Napisz program rysujący kopertę:



Z użyciem grafiki żółwia.

REKURENCJA

Rekurencja zachodzi wtedy, gdy jakaś procedura lub funkcja wywołuje samą siebie.

PRZYKŁAD 32

Narysować płot o zadanej długości (liczbie sztachet).

Zdefiniuj procedurę SZTACHETA z trzema parametrami:

N liczba sztachet do narysowania

X,Y współrzędne pierwszej sztachety

PROGRAM REK2;

USES

GRAPH,CRT,ZOLW;

VAR

TRYB,KARTA:INTEGER;

PROCEDURE SZTACHETA(N,X,Y:INTEGER);

BEGIN

IF N=0 THEN

BEGIN

REPEAT UNTIL KEYPRESSED;

HALT;

```

END;
kat:=0;
MOVETO(X,Y);
NP(4);LEWO(90);NP(40);
LEWO(60);NP(4);LEWO(60);
NP(4);LEWO(60);NP(40);
SZTACHETA(N-1,X+10,Y+3);
END;
BEGIN
KARTA:=DETECT;
INITGRAPH(KARTA,TRYB,'C:\TP\BGI');{ tutaj twoja ścieżka}
INICJUI_ZOLWIA;
SZTACHETA(20,10,10);
REPEAT UNTIL KEYPRESSED;
CLOSEGRAPH;
END.

```

ZADANIE 76.

Po wpisaniu programu REKU2 wykonaj:

- 1) zmień liczbę rysowanych sztachet na numer w dzienniku + 2
- 2) zmień odległość między sztachetami
- 3) narysuj płot w poziomie

W zeszycie przepisz tylko zmienione linijki programu, które realizują polecenie 1), 2), 3)

Np.

- 1)

..... tuta linia lub linie programu
- 2)

..... tuta linia lub linie programu
- 3)

..... tuta linia lub linie programu

ZADANIE 77.

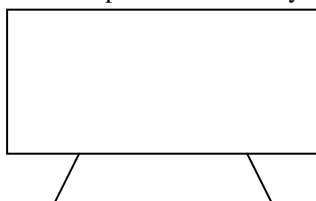
Narysować przy wykorzystaniu rekurencji układ telewizora oraz patrzącej na nią kamery.

Zdefiniować procedurę graficzną TELEWIZOR z trzema parametrami:

x,y współrzędne początku rysowania telewizora

w wymiar telewizora

Zapewnić przerwanie programu gdy wymiar telewizora jest mniejszy od 5. Wszystkie linie z ,których powstaje telewizor powinny być uzależnione od parametru w. Przybliżony(proponowany) wygląd telewizora:



ZADANIE 78.

Rekurencyjne przedstawianie ciągu.

Ciągi liczbowe mogą być przedstawione w postaci rekurencyjnej. Rekurencyjny sposób przedstawiania ciągów polega na tym, że następny wyraz ciągu znajdujemy na podstawie znajomości poprzedniego.

$$\begin{cases} a_1 = 2 \\ a_{n+1} = a_n + 2^n \end{cases}$$

$$\begin{cases} n = 1 \\ a_{1+1} = a_1 + 2^1 = 2 + 2 = 4 \end{cases}$$

$$\begin{cases} n = 2 \\ a_{2+1} = a_2 + 2^2 = 4 + 4 = 8 \end{cases}$$

$$\begin{cases} n = 3 \\ a_{3+1} = a_3 + 2^3 = 8 + 8 = 16 \end{cases}$$

$$\begin{cases} n = 4 \\ a_{4+1} = a_4 + 2^4 = 16 + 16 = 32 \end{cases}$$

Oblicz a_5 , pamiętaj, że będziesz musiał obliczyć wszystkie poprzednie wyrazy a_2 a_3 a_4

$$\begin{cases} a_1 = 4 \\ a_{n+1} = a_n * 1,5 \end{cases}$$

Sinia

W matematyce definiowana jest pojęcie silni np. $n!$

Def:

$0! = 1$

$1! = 1$

.....

$5! = 1 * 2 * 3 * 4 * 5 = 120$

Oblicz $7!$

Za pomocą silni możemy np. obliczyć na ile sposobów można ustawić pięć osób w kolejce w sklepie, czyli $5! = 120$.

Symbol Newtona

Do wykonywania pewnych obliczeń związanych z podawaniem ilości kombinacji stosowany jest symbol Newtona.

Np. gdy chcemy obliczyć ile jest wszystkich możliwych kuponów w totolotku(sześć liczb z czterdziestu dziewięciu) można zastosować symbol Newtona $K=6$ $N=49$.

$$\binom{N}{K} = C_N^K = \frac{N!}{(N-K)! * K!}$$

np.

$$\binom{5}{3} = C_5^3 = \frac{5!}{(5-3)! * 3!} = \frac{1 * 2 * 3 * 4 * 5}{1 * 2 * 1 * 2 * 3} = 10$$

Oblicz

$$\binom{7}{4} = ?$$

PRZYKŁAD 33

Obliczyć wartość silni stosując rekurencję. Silnia w sposób rekurencyjny jest definiowana:

$$n! = \begin{cases} 1 & \text{dla } n = 0 \\ n(n-1)! & \text{dla } n > 0 \end{cases}$$

Wykonaj:

a)zapisz definicję rekurencyjną silni

b)sprawdź i zapisz w zeszytce do jakiej wielkości można obliczać silnię z użyciem tego przykładu.

c)wpisz przykład do zeszytu.

PROGRAM REKU3;

USES

CRT;

VAR

A:REAL;

M:INTEGER;

FUNCTION SILNIA(N:INTEGER):REAL;

BEGIN

IF N=0 THEN

SILNIA:=1

```

ELSE
  SILNIA:=SILNIA(N-1)*N;
END;
BEGIN
  CLRSCR;
  WRITE('podaj n=');
  READLN(M);
  A:=SILNIA(M);
  WRITELN('SILNIA('M,')=',A:6:1);
  REPEAT UNTIL KEYPRESSED;
END.

```

ZADANIE 79.

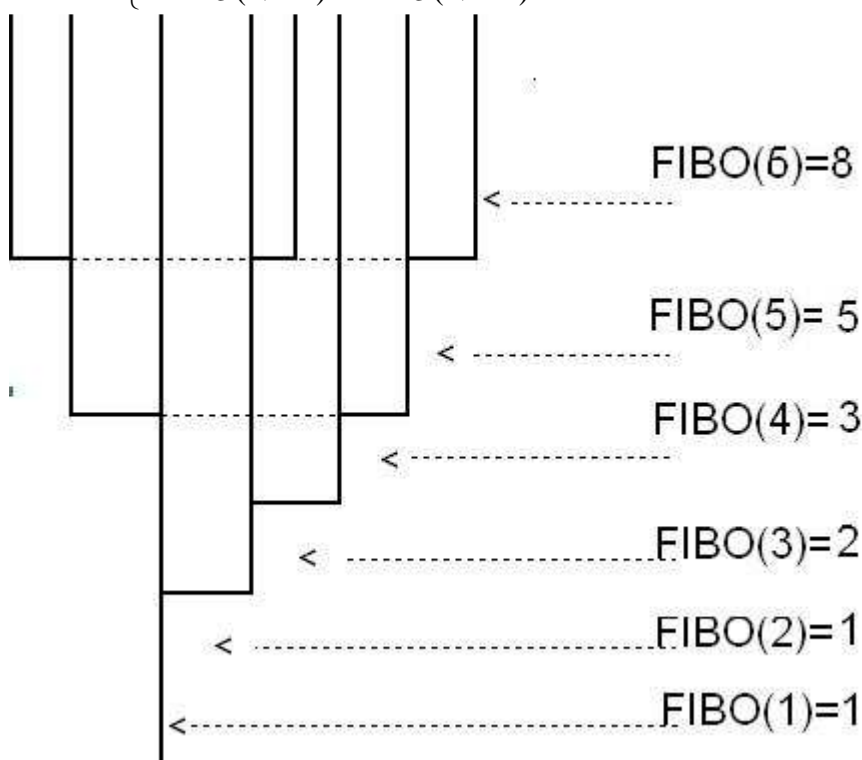
Wykorzystując funkcję SILNIA zapisz funkcję NEWTON z dwoma parametrami N i K, która będzie obliczać symbol Newtona

$$\binom{N}{K} = \frac{N!}{(N-K)! * K!}$$

ZADANIE 80.

Napisz program obliczający dowolne wyrazy ciągu Fibonaciego. Program obliczać będzie również ilość pędów drzewa w N-tym roku. Rekurencyjna definicja ciągu Fibonaciego jest następująca:

$$FIBO = \begin{cases} 1 & \text{DLA } N = 1 \text{ LUB } N = 2 \\ FIBO(N-1) + FIBO(N-2) \end{cases}$$



Liczby Fibonaciego FIBO(N) pojawiają się przy opisie ilościowym niektórych zjawisk przyrodniczych. Jeśli rozrastanie roślin odbywa się zgodnie z zasadą: każdy pęd wypuściwszy pęd boczny, przez rok odpoczywa i dopiero w następnym roku puszcza nowy pęd, to liczba pędów w N-tym roku może być obliczona zgodnie z wcześniej podanym ciągiem rekurencyjnym FIBO.

Schemat Hornera

Schemat Hornera jest algorytmem służącym do:

- szybkiego obliczania wartości wielomianów,
- przeliczanie na system dziesiętny liczb zapisanych w innym systemie liczbowym,
- szybkie podnoszenie do potęgi.

Opis schematu Hornera

Wielomian stopnia trzeciego zapisany w sposób klasyczny to:

$$W(x) = a_0 \cdot x^3 + a_1 \cdot x^2 + a_2 \cdot x + a_3$$

Można zapisać inaczej w postaci iloczynowej

$$W(x) = ((a_0 \cdot x + a_1) \cdot x + a_2) \cdot x + a_3$$

Ocena złożoności obliczeń:

Aby obliczyć wartość wielomianu zapisanego w sposób klasyczny trzeba sześciu mnożeń (tyle mnożeń jest konieczne w celu obliczenia potęgi) oraz trzech dodawań. W algorytmie Hornera konieczne są trzy mnożenia i trzy dodawania. W roku 1971 udowodniono, że schemat Hornera jest najszybszym algorytmem obliczania wartości wielomianu.

ZADANIE 81.

a) Zapisz w zeszyście do czego służy schemat Hornera.

b) Zapisz w sposób klasyczny oraz schematem Hornera wielomian czwartego stopnia.

Dane:

a_0 =numer z dziennika, a_1 =dzień urodzenia, a_2 =miesiąc urodzenia, a_3 =liczba liter imienia, a_4 =(liczba liter imienia+2).

→ sposób klasyczny (zapisz w zeszyście)

→ sposób iloczynowy (zapisz w zeszyście)

c) oblicz dwoma sposobami $W(-1)$

d) Oceń i zapisz złożoność obliczeń dla wielomianu stopnia=numer z dziennika+6

Uogólnienie schematu Hornera

$$W(x) = a_0 \cdot x^n + a_1 \cdot x^{n-1} + \dots + a_{n-1} \cdot x + a_n \quad \text{dla } n \geq 0$$

$$W(x) = (\dots(((a_0 \cdot x + a_1) \cdot x + a_2) \cdot x + a_3) \cdot x + \dots + a_{n-1} \cdot x + a_n)$$

Różne zapisy schematu Hornera

Postać iteracyjna	Postać rekurencyjna
$W(x) := a_0 \Rightarrow \text{wart początkowa}$ $W(x) = W(x) \cdot x + a_i \text{ dla } i = 1, 2, \dots, n$	$W_n(x) = \begin{cases} a_0 \\ W_{n-1}(x) \cdot x + a_n \end{cases}$

Rekurencyjny sposób obliczania wielomianu czwartego stopnia dla argumentu $x=k$.

$$W_4(k) := W_3(k) \cdot k + a_4$$

$$W_3(k) := W_2(k) \cdot k + a_3$$

$$W_2(k) := W_1(k) \cdot k + a_2$$

$$W_1(k) := W_0(k) \cdot k + a_1$$

$$W_0(k) := a_0$$

Czyli aby obliczyć wartość wielomianu czwartego stopnia musimy obliczyć wartość wielomianu trzeciego stopnia aby obliczyć wartość wielomianu trzeciego stopnia musimy obliczyć wartość wielomianu drugiego stopnia itd. W ten sposób tworzymy ciąg wywołań rekurencyjnych czyli **stos**.

ZADANIE 82.

a) Zapisz w zeszyście schemat Hornera w postaci iteracyjnej.

b) Zapisz w zeszyście schemat Hornera w postaci rekurencyjnej

c) zapisz rekurencyjny sposób obliczania wielomianu czwartego stopnia dla argumentu $x=k$ (pamiętaj o wcięciach).

d) Zapisz w zeszyście wywołania rekurencyjne dla wielomianu czwartego stopnia dla danych jak w zadaniu poprzednim (czyli stosuj: $a_0, a_1, a_2, a_3, a_4, a_5$ z zadania poprzedniego) oraz k =liczba liter imienia. Zastosuj wcięcia jak w przykładzie powyżej. Zauważ, że obliczenia wykonujemy od dołu.

e) zapisz w zeszyście co to jest stos i na czym polega jego budowa.

ZADANIE 83.

Napisz program obliczający wartość wielomianu metodą iteracyjną.

Specyfikacja

Dane:

tablica współczynników wielomianu

→ $A[a_0, \dots, a_n]$

stopień wielomianu $\rightarrow n$
wartość argumentu dla którego liczymy wartość wielomianu $\rightarrow k$

Wynik:

wartość wielomianu dla danych współczynników czyli $W(k)$

Zarezerwuj tablicę jako 20 elementową dla liczb rzeczywistych od elementu o indeksie zero. Czyli ostatni jest $a[19]$ a pierwszy $a[0]$.

Wykonaj schemat blokowy.

ZADANIE 84.

Napisz program obliczający wartość wielomianu metodą rekurencyjną.

Specyfikacja

Dane:

tablica współczynników wielomianu $\rightarrow A[a_0, \dots, a_n]$

stopień wielomianu $\rightarrow n$

wartość argumentu dla którego liczymy wartość wielomianu $\rightarrow k$

Wynik:

wartość wielomianu dla danych współczynników czyli $W(k)$

Zarezerwuj tablicę jako 20 elementową dla liczb rzeczywistych od elementu o indeksie zero. Czyli ostatni jest $a[19]$ a pierwszy $a[0]$.

ZADANIE 85. (na ocenę bardzo dobrą)

Wartości funkcji elementarnych, takich jak \sin , \cos , \log , są obliczane za pomocą komputera w sposób przybliżony. Często stosuje się w tym celu wzory, które mają postać nieskończonych sum. Na przykład prawdziwy jest następujący wzór na wartość logarytmu naturalnego z liczby 2:

$$\ln 2 = \frac{2}{3} \left(1 + \frac{1}{3} \cdot \frac{1}{9} + \frac{1}{5} \cdot \frac{1}{9^2} + \frac{1}{7} \cdot \frac{1}{9^3} + \frac{1}{9} \cdot \frac{1}{9^4} + \frac{1}{11} \cdot \frac{1}{9^5} + \dots \right)$$

W oparciu o powyższy wzór można zaprojektować i napisać program, który dla danej liczby ε ($\varepsilon > 0$) oblicza przybliżoną wartość $\ln 2$, sumując jak najmniej wyrazów, aby różnica między dwoma ostatnimi przybliżeniami była mniejsza niż ε .

Wprowadźmy oznaczenie:

dla $n \geq 1$

$$l_n = \frac{2}{3} \left(1 + \frac{1}{3} \cdot \frac{1}{9} + \frac{1}{5} \cdot \frac{1}{9^2} + \frac{1}{7} \cdot \frac{1}{9^3} + \dots + \frac{1}{2n+1} \cdot \frac{1}{9^n} \right)$$

$$l_0 = \frac{2}{3}$$

Wykonaj poniższe polecenia:

Wypełnij tabelę:

n	L_n
0	
1	
2	
3	

Poniżej podaj zależność pomiędzy wartościami l_n i l_{n-1} dla każdego $n=1, 2, \dots$

Podaj wzór rekurencyjny na różnicę $r_n = l_n - l_{n-1}$ dla $n > 0$:

b) Podaj algorytm ze specyfikacją (w postaci listy kroków, schematu blokowego lub w języku programowania), który dla danej liczby ε ($\varepsilon > 0$) oblicza przybliżoną wartość $\ln 2$, sumując jak najmniej wyrazów we wzorze podanym w treści zadania, aby różnica między dwoma ostatnimi przybliżeniami była mniejsza niż ε .

Fraktale

Fraktalami nazywamy figury geometryczne wykazujące podobieństwo w swojej budowie niezależnie od skali w jakiej są obserwowane. Znalazły zastosowanie między innymi w fizyce, biologii, grafice komputerowej. Fraktal definiowany jest przez :

–stopień


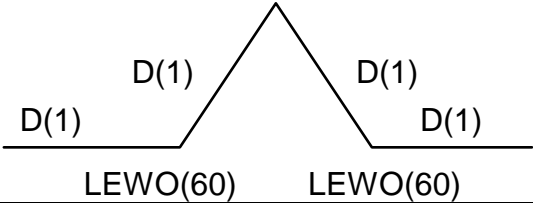
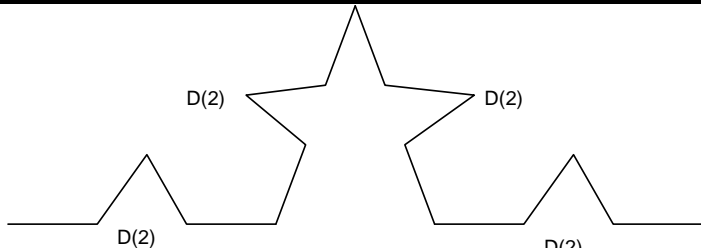
–kształt

–długość

PRZYKŁAD 34

Napisać program rysujący płatek KOCHA.

Przerysuj tabelkę do zeszytu pis oraz wzór rekurencyjny.:

opis fraktala	wygląd fraktala	stopień fraktala
D(1)=np(a) a- długość elementarna odcinka fraktala	odcinek o długości a 	D(1) oznacza że jest to fraktal pierwszego stopnia
D(2)		D(2) oznacza że jest to fraktal drugiego stopnia
D(3)		D(3) oznacza że jest to fraktal trzeciego stopnia

Mamy tu do czynienia z przekształceniem odcinka D(1) polegającym na dorysowaniu w jego środku części dwóch boków trójkąta równobocznego. Powstaje w ten sposób figura D(2), złożona z czterech odcinków. W następnym kroku każdy z tych odcinków przekształcamy analogicznie i otrzymujemy fraktal 3 stopnia D(3). Cały płatek Kocha składa się z trzech figur D(n) opartych na bokach trójkąta równobocznego (trzy figury obrócone w prawo o 120 stopni).

Rekurencyjną formułę generowania płatka Kocha można ująć:

$$\begin{cases} D(1) = np(a); \\ D(k) = D(k-1); LEWO(60); D(k-1); PRAWO(120); D(k-1); LEWO(60); D(k-1); \end{cases}$$

gdzie k jest stopniem fraktala

```

PROGRAM REKU5;
USES
  ZOLW,GRAPH,CRT;
VAR
  I:INTEGER;
  TRYB,STEROWNIK:INTEGER;
PROCEDURE D(K:INTEGER);
BEGIN
  IF K=1 THEN
    NP(8)
  ELSE
    BEGIN
      D(K-1);LEWO(60);
      D(K-1);PRAWO(120);
      D(K-1);LEWO(60);
      D(K-1);
    END;
  END;
END;
BEGIN
  STEROWNIK:=DETECT;
  INITGRAPH(STEROWNIK,TRYB,'C:\TP\BGI');
  MOVETO(GETMAXX DIV 2,GETMAXY DIV 2);
  FOR I:=1 TO 3 DO
    BEGIN
      D(4);PRAWO(120);
    END;
  REPEAT UNTIL KEYPRESSED;
  CLOSEGRAPH;

```

END.

ZADANIE 86.


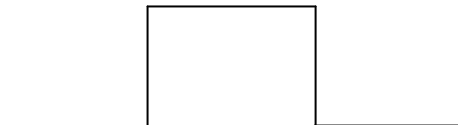
Zmień długość rysowania odcinka podstawowego z 4 na taką wielkość aby przy zmianie stopnia rysowania na 5 a następnie na 6 fraktal zmieścił się na ekranie. Jeśli istnieje konieczność to zmień również punkt startowy rysowania fraktala. Zapisz w zeszycie zmienioną linię programu.

ZADANIE 87.

W celu otrzymania fraktala zwanego KWIATKIEM należy odwrócić figurę $D(k)$ do wewnątrz trójkąta (trzy figury obrócone w lewo o 120 stopni). Zmień tylko jedną instrukcję w programie poprzednim a otrzymasz KWIATEK. Zapisz w zeszycie zmienioną linię programu.

ZADANIE 88.

Napisz program do generowania fraktala zdefiniowanego w następujący sposób:

	
$D(1)=np(a)$ Pierwszy stopień fraktala	$D(2)$ drugi stopień fraktala

Przed napisaniem programu wykonaj w zeszycie:

–narysuj fraktala trzeciego stopnia

–napisz wzór rekurencyjny tworzenia fraktala k -tego stopnia

Fraktal powinien składać się z czterech figur $D(k)$ tworzących kwadrat.

ZADANIE 89.

W punktach a) b) c) nazwa funkcji to CIAG_nazwisko_ucznia np. CIAG_kowalski

a)

Dany jest ciąg liczbowy $a_1=3$ $a_2=7$ $a_3=16$ $a_4=32$ $a_5=57$ $a_6=93$..

Napisz w zeszycie wzór rekurencyjny obliczający n -ty wyraz ciągu. Napisz na podstawie wzoru rekurencyjnego program obliczający n -ty wyraz ciągu. Oblicz 17-ty wyraz tego ciągu. Zapisz program w zeszycie.

b)

Dany jest ciąg liczbowy $a_1=2$ $a_2=7$ $a_3=12$ $a_4=17$ $a_5=22$ $a_6=27$..

Napisz w zeszycie wzór rekurencyjny obliczający n -ty wyraz ciągu. Napisz na podstawie wzoru rekurencyjnego program obliczający n -ty wyraz ciągu. Oblicz 20-ty wyraz tego ciągu. Zapisz funkcję rekurencyjną pascala w zeszycie.

c)

Dany jest ciąg liczbowy $a_1=1$ $a_2=9$ $a_3=36$ $a_4=100$ $a_5=225$..

Napisz w zeszycie wzór rekurencyjny obliczający n -ty wyraz ciągu. Napisz na podstawie wzoru rekurencyjnego program obliczający n -ty wyraz ciągu. Oblicz 7-ty wyraz tego ciągu. Zapisz funkcję rekurencyjną pascala w zeszycie.

ZADANIA PRZYKŁADOWE NA SPRAWDZIAN

ZADANIE 1

Napisz moduł, który będzie zawierał następujące :

–funkcje

a)obliczanie pola trójkąta równobocznego

b)obliczania obwodu trójkąta równobocznego

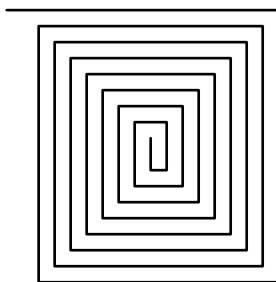
–procedury

a)rysowania trójkąta równobocznego ze zmienną długość boku

Napisz również program, który będzie testował moduł sprawdzając poprawność wszystkich procedur i funkcji tego modułu.

ZADANIE 2

Wykorzystując rekurencję oraz grafikę żółwia napisz program



rysujący daną figurę
Kolejne boki powstają przez
obrót o 90 stopni i dodanie
stałej d do długości poprzed-
niego boku. Rysowanie zaczynamy
od środka. Program powinien
zakończyć rysowanie gdy wymiar
najdłuższego boku jest większa
od 200 pikseli.

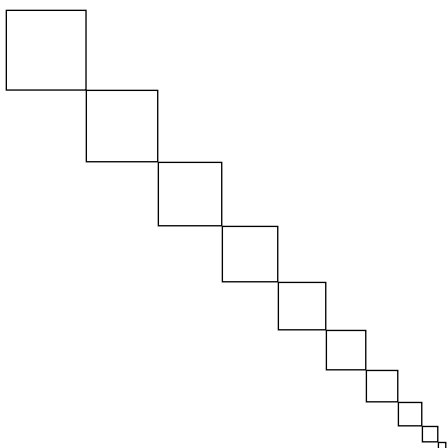
ZADANIE 3

Napisz program rysujący fraktala zwanego smokiem Hartnera. Smok Hartnera jest fraktalem składającym się z tzw. smoka prawego i smoka lewego

Stopień	Smok lewy	Smok prawy
Stopień 0		
Stopień 1		
Stopień 2		

Aby narysować smoka $n+1$ stopnia trzeba do każdego boku smoka n przybudować równoramienny trójkąt prostokątny, raz po prawej stronie a raz po lewej stronie tej łamanej. Czyli każdy smok prawy n -tego stopnia składa się z smoka prawego $n-1$ stopnia oraz obroconego (musisz wywnioskować z rysunku o jaki kąt i w jaką stronę) smoka lewego. Każdy smok lewy n -tego stopnia składa się z smoka prawego $n-1$ stopnia oraz obroconego (musisz wywnioskować z rysunku o jaki kąt i w jaką stronę) smoka lewego. Ważny jest kąt startowy tego fraktala dla stopnia 0 jest to zero stopni dla stopnia 1 jest to 45 stopni dla stopnia 2 jest 90 stopni itd. czyli wartość początkowa kąta zależy od stopnia musisz to uwzględnić w programie. Kolejne boki fraktala dla coraz większych stopni zmniejszają się (oblicz o jaką liczbę i uwzględnij w programie). Procedur rysując fraktal powinna mieć dwa parametry stopień oraz bok. Program powinien być tak napisany aby pytał się o stopień oraz bok zapewnij również powtarzalność.

ZADANIE 4



Narysować za pomocą rekurencji sześć kwadratów o zmniejszającym się boku o trzy piksele. Kwadraty muszą być ułożone jak na rysunku.

Użyj:

PROCEDURE(X,Y,W:INTEGER);

X,Y– współrzędne początku rysowania

W– wymiar kwadratu

ZADANIE 5

Algorytm Newtona–Raphsona obliczania pierwiastka kwadratowego z liczby A wygląda następująco

$$x_i = 0.5 \left(x_{i-1} + \frac{A}{x_{i-1}} \right)$$

gdzie: A – liczba z której szukamy pierwiastka

x_i – kolejne przybliżenia pierwiastka.

Na podstawie tego iteracyjnego wzoru można zapisać rekurencyjny wzór znajdowania pierwiastka kwadratowego (podobnie jak wzór na obliczanie $n!$). W zeszycie zapisz ten wzór. Wzorując się na programie na obliczanie $n!$ napisz procedurę PIERW(N:INTEGER); gdzie N ilość kolejnych przybliżeń przy obliczaniu pierwiastka. Funkcja PIERW będzie obliczała pierwiastek z np. 9. W procedurze należy wybrać pierwsze przybliżenie czyli $x(1)$ np. $x(1)=1$. Program będzie pytał się o liczbę przybliżeń a następnie wywoływał funkcję PIERW. Program będzie miał możliwość powtarzania. Zbadaj wpływ pierwszego przybliżenia na dokładność obliczeń (przy stałej liczbie przybliżeń). Zbadaj wpływ ilość przybliżeń (przy stałej wartości początkowej) na dokładność obliczeń.

ilość zadań ocena:

0–1	niedostateczna
2	dop
3	dostateczna
4	dobra
5	bardzo dobra

Część ósma wykładu z PASCALA

Operacje na plikach

Rodzaje plików:

- Pliki tekstowe

Są to zbiory danych zapisywane w postaci ciągu znaków. Mogą być tworzone, zapisywane i odczytywane za pomocą dowolnego edytora tekstowego.

- Pliki elementowe

Są to zbiory danych zapisywane w zakodowanej binarnej formie. Niemożliwe do odczytania za pomocą edytora tekstowego.

Porównanie plików elementowe (sekwencyjnych) z tekstowymi

a) w plikach elementowych (*są to pliki o dostępie swobodnym*) masz dostęp do dowolnego elementu

a w plikach tekstowych należy przeczytać wszystko co jest przed nim aby uzyskać dostęp do dowolnego elementu,

b) pliki elementowe wykorzystują reprezentacje zakodowane nie da się ich odczytać za pomocą edytora tekstowego.

Możesz zapisywać elementy tylko tego samego typu w jednym pliku.

pliki tekstowe można formatować wydruki i wykorzystywać instrukcje writeln oraz readln. Możesz zapisywać elementy różnych typów w jednym pliku.

c) w plikach tekstowych masz do dyspozycji jeszcze jedną instrukcję **append**

append(nazwa_zmiennej_plikowej);

otwarcie pliku tekstowego do tzw. dopisywania, wskaźnik ustawiany jest na końcu pliku i teraz możesz dopisywać do pliku tekstowego na jego koniec

Instrukcje stosowane przy działaniach na plikach

assign(nazwa_zmiennej_plikowej,'nazwa_pliku');

skojarzenie nazwy zmiennej plikowej w programie ze zbiorem istniejącym na dysku. Nazwa pliku na dysku może być poprzedzona ścieżką dostępu do niego

reset(nazwa_zmiennej_plikowej);

otwarcie już istniejącego pliku (gdy go nie będzie to będzie sygnalizowany błąd) i ustawienie wskaźnika pliku na pierwszy element. Wskaźnik pliku jest po każdej operacji zapis/odczyt tak modyfikowany, że wskazuje następny element pliku(można ten wskaźnik dowolnie ustawiać)

rewrite(nazwa_zmiennej_plikowej);

otwieranie pliku niezależnie od tego czy istniał czy nie. Jeśli istniał taki plik to możemy liczyć się z utratą danych z niego.

close(nazwa_zmiennej_plikowej);

zamknięcie pliku otwartego przez reset lub rewrite. Nie zamknięcie pliku powoduje utratę danych.

write(nazwa_zmiennej_plikowej,zmienne);

zapisanie wartości zmiennej do pliku

read(nazwa_zmiennej_plikowej,zmienne);

odczytanie wartości zmiennej z pliku

uwaga: niedozwolone jest stosowanie writeln oraz readln przy plikach elementowych

eofl(nazwa_zmiennej_plikowej)

sprawdzanie czy koniec pliku. Jeśli tak to zwraca true

seek(nazwa_zmiennej_plikowej,numer_elementu);

ustawienie wskaźnika na element o podanym numerze

uwaga: elementy są numerowane od zera dlatego gdy chcemy uzyskać dostęp do 20 elementu to wykonujemy seek(f,19);

filesize(nazwa_zmiennej_plikowej);

podaje ilość elementów pliku

filepos(nazwa_zmiennej_plikowej);

podaje aktualną pozycję wskaźnika

Uzyskiwanie dostępu do grupy pól określonego rekordu

Dostępu do określonego pola rekordu można uzyskać stosując instrukcję **WIAŻACA WITH** Zastosowanie tej instrukcji pozwoli na wyeliminowanie odwołania się do pola rekordu poprzez pisanie jego nazwy a następnie po kropce nazwy pola.

WITH REKORD DO

np.

WITH OSOBA[10] DO

BEGIN

NAZWISKO:='KOWALSKI';

IMIE :='MARIAN';

```

      WAGA      :=78;
END;

```

Nastąpiło wpisanie do rekordu dziesiątego o nazwie OSOBA do pól NAZWISKO IMIĘ WAGA

PLIKI ELEMENTOWE

PRZYKŁAD 35

Zapisz w zeszycie znaczenie następujących pojęć:

- Rodzaje plików:
- Porównanie plików sekwencyjnych z tekstowymi
- Instrukcje stosowane przy działaniach na plikach

Treść przykładu praktycznego.

Jeśli istnieje potrzeba zabezpieczenia programu hasłem można to zrobić w prosty sposób. Zabezpieczenie (program), które ma zapisane hasło na zewnątrz programu jest jednak bardzo łatwe do złamania. Wystarczy program z rozszerzeniem EXE przeglądnąć w dowolnym edytorze tekstowym i hasło może być odczytane. Dlatego stosuje się zapisywanie haseł w osobnych plikach. Pliki z hasłami mogą być następnie kodowane.

Poniżej zostaną przedstawione dwa programy:

–do zapisywania hasła w pliku PROGRAM PLIKI1

–do odczytywania hasła z pliku PROGRAM PLIKI2

Zbiór z hasłami zostanie zapisany na dysku C w folderze cztery_pierwsze_litery_twojego_nazwiska (utwórz taki folder) a jego nazwa będzie numer_z_dziennika_haslo.dat. Hasło podaj Twoje nazwisko.

- Wpisz oba programy i sprawdź czy działają,
- przeglądaj plik numer_z_dziennika_haslo.dat w dowolnym edytorze, i w tym momencie poproś nauczyciela w celu zaliczenia tak aby mógł odczytać listing programów (w zeszycie) oraz plik numer_z_dziennika_haslo.dat.

```

PROGRAM PLIKI1;
USES
  CRT;
VAR
  HASLO:STRING;
  ZBIOR_Z_HASLEM : FILE OF STRING;
BEGIN
  CLRSCR;
  WRITE('PODAJ HASŁO=');
  READLN(HASLO);
  CLRSCR;
  ASSIGN(ZBIOR_Z_HASLEM,'C:\cztery_litery_nazwiska\37_haslo.dat'); {37 zamień na inny numer}
  REWRITE(ZBIOR_Z_HASLEM);
  WRITE(ZBIOR_Z_HASLEM,HASLO);
  CLOSE(ZBIOR_Z_HASLEM);
END.

```

```

PROGRAM PLIKI2;
USES
  CRT;
VAR
  HASLO:STRING;
  HASLO_WCZYT:STRING;
  ZBIOR_Z_HASLEM : FILE OF STRING;
BEGIN
  CLRSCR;
  ASSIGN(ZBIOR_Z_HASLEM,'37_haslo.dat'); {37 zamień na inny numer oraz ścieżkę dostępu}
  RESET(ZBIOR_Z_HASLEM);
  READ(ZBIOR_Z_HASLEM,HASLO);
  CLOSE(ZBIOR_Z_HASLEM);
  WRITE('PODAJ HASŁO=');
  READLN(HASLO_WCZYT);
  IF HASLO=HASLO_WCZYT THEN
    WRITELN('BRAVO ZNASZ HASŁO')
  ELSE
    WRITELN('ŻEGNAJ NIE ZNASZ HASŁA');
  REPEAT UNTIL KEYPRESSED;

```

END.

PRZYKŁAD 36

Na dysku można zapisywać całe tablice. Odczytywać można również całe tablice lub element po elemencie. Poniżej zostanie zaprezentowany program zapisywania tablicy liczbami z przedziału $<0,9>$. Następnie cała ta tablica zostanie zapisana na dysku. Odczytywanie odbywać się będzie jednak element po elemencie.

```
PROGRAM PLIKI3;
USES
  CRT;
TYPE
  TABLICA=ARRAY[1..100] OF REAL;
VAR
  ZBIOR_Z_TABLICA : FILE OF TABLICA;
  DANE,DANE_PO_WCZYT: TABLICA;
  WCZYT_LICZBY : FILE OF REAL;
  I:INTEGER;
  ELEMENT:REAL;
BEGIN
  { ---- WYLOSOWANIE TABLICY STU ELEMENTÓW }
  { ----- ORAZ JEJ ZAPISANIE NA DYSK ---- }
  { -----JAKO CAŁEJ TABLICY ----- }
  CLRSCR;
  RANDOMIZE;
  ASSIGN(ZBIOR_Z_TABLICA,'A:TABLICA.DAT'); {podaj pełna ścieżkę dostępu dla twoich plików}
  REWRITE(ZBIOR_Z_TABLICA);
  FOR I:=1 TO 100 DO
    DANE[I]:=RANDOM(10);
  WRITE(ZBIOR_Z_TABLICA,DANE);
  CLOSE(ZBIOR_Z_TABLICA);

  { -----WCZYTANIE DANYCH Z DYSKU ----- }
  {-WCZYTYWANIE PO JEDNYM WYRAZIE NIE CAŁA TABLICA-}

  ASSIGN(WCZYT_LICZBY,'A:TABLICA.DAT');
  RESET(WCZYT_LICZBY);
  I:=0;
  WHILE NOT EOF(WCZYT_LICZBY) DO
  BEGIN
    INC(I); {ZWIĘKSZENIE I O JEDEN }
    READ(WCZYT_LICZBY,DANE_PO_WCZYT[I]);
    WRITELN('POZYCJA W ZBIORZE DANYCH ',FILEPOS(WCZYT_LICZBY));
    WRITELN(I,' ',DANE_PO_WCZYT[I]:6:2);
    DELAY(500);
    END;
    WRITE('PODAJ ,KTÓRY ELEMENT ZBIORU CHCESZ ODCZYTAĆ NR=');
    READ(I);
    SEEK(WCZYT_LICZBY,I-1);
    READ(WCZYT_LICZBY,ELEMENT);
    WRITELN('ELEMENT ',I,' JEST ',ELEMENT:6:2);
    CLOSE(WCZYT_LICZBY);
  REPEAT UNTIL KEYPRESSED;
END.
```

ZADANIE 90.

- Wylosuj tablicę $200+nr_z_dziennika$ elementową z przedziału $<2,10+miesiąc_urodzenia>$,
- zapisz tablicę jak plik elementowy na dyskietkę ucznia po nazwę numer_z_dziennika_ele.dat,
- używając instrukcji SEEK oblicz sumę wyrazów nieparzystych np. $A[1]+A[3]+.....$ a następnie parzystych, $A[2]+A[4]+.....$
- zapisz w zeszycie specyfikację problemu,
- przeglądaj plik numer_z_dziennika_ele.dat w dowolnym edytorze,
- poproś nauczyciela w celu zaliczenia tak aby mógł odczytać listing programu oraz plik numer_z_dziennika_ele.dat.

ZADANIE 91.

Za wykonanie tego zadania dostaniesz ocenę do dziennika.

Poroś nauczyciela o program w postaci pliku *.pas. Uruchom ten program. Wpisz trzy osoby. Wykonaj próbę wszystkich opcji.

Na podstawie treści programu otrzymanego od nauczyciela. Napisz bazę danych na zadany temat:

W tabeli poniżej jest numer w dzienniku oraz temat bazy danych.

Założenia do bazy danych:

- baza jest jedno_tabelaryczna
- nazwa bazy danych (w pamięci komputera)→baza_cztery_pierwsze_litery_nazwisk,
- nazwa bazy danych (w dysku)→baza_cztery_pierwsze_litery_nazwisk.dat,
- nazwa rekordu →rekord_cztery_pierwsze_litery_nazwisk,
- nazwa pola rekordu→znacząca_nazwa_dwie_pierwsze_litery_nazwisk np. waga_ko
- Opcje menu
 - Wypisanie danych
 - Dopisywanie danych
 - Pierwsze wprowadzanie danych-kasuje poprzednie
 - Wyszukiwanie danych w/g
 - Wyszukiwanie danych w/g
 - Wyszukiwanie danych w/g
 - Pomoc (temat pracy, autor, struktura rekordu, użycie kolorów podczas pisania tekstu pomocy)
 - Sortowanie(na 6→ pytanie o pole w, którego chcemy sortować+wyświetlenie po sortowaniu, sygnały dźwiękowe przy przechodzeniu między opcjami menu, ramki z kodów ASCII dla menu oraz wyświetlanych rekordów.)
 - Wyjście

nr	Temat Bazy	nr	Temat Bazy
1	Hurtownia sprzętu elektrycznego	16	Hurtownia sprzętu elektrycznego
2	Obsługa salonu sprzedaży samochodów	17	Narzędziownia w stoczni.
3	System bankowy, udzielanie kredytów	18	System rezerwacji i sprzedaży biletów w kinie
4	Obsługa sekretariatu uczniowskiego	19	System Obsługi Przychodni lekarskiej
5	Hurtownia kosmetyków	20	Obsługa salonu sprzedaży samochodów
6	Sprzedaż sprzętu komputerowego (sklep)	21	Sprzedaż sprzętu komputerowego (sklep)
7	System Obsługi Przychodni lekarskiej	22	Biuro obsługi mieszkańców w spółdzielni mieszkaniowej
8	System obsługi apteki	23	Stacja naprawy samochodów
9	System rezerwacji i sprzedaży biletów w kinie	24	Salon oraz komis sprzedaży samochodów
10	Biuro obsługi mieszkańców w spółdzielni mieszkaniowej	25	Sekretariat skoczka narciarskiego
11	Salon oraz komis sprzedaży samochodów	26	System obsługi Lombardu
12	Stacja naprawy samochodów	27	Płace w przedsiębiorstwie
13	Obsługa sekretariatu uczniowskiego	28	Wypożyczalnia kaset oraz płyt CD
14	System obsługi apteki	29	Stacja meteorologiczna
15	System bankowy, udzielanie kredytów	30	System obsługi przedszkola

Program baza_uczniow;

uses crt;

type

OpisUcznia = record

Imie : string[15];

Nazwisko : string[20];

O_polski : integer;

O_matma : integer;

```

    Godziny_u: integer;
    Godziny_n: integer;
end;
var
    klasa : array[1..40] of OpisUcznia;
    BazaDanych : file of OpisUcznia;
    y:byte;
    c1,c2:char;
    dalej:char;
    tablica:array[1..10] of string;

procedure menu;forward;

procedure wprowadz_dane; {procedura wprowadzania danych, pierwszy raz}
var i:byte;
begin
    clrscr;
    dalej:=readkey;
    assign(BazaDanych,'dziennik.dat');
    rewrite(BazaDanych);      {otwieram plik po raz 1 }
    i:=0;
    Repeat
        i:=i+1;
        writeln;
        write('Imie: ');      readln(klasa[i].imie);
        write('Nazwisko: ');  readln(klasa[i].nazwisko);
        write('Oena z Języka Polskiego: '); readln(klasa[i].O_polski);
        write('Ocena z matmy: '); readln(klasa[i].O_matma);
        write('Ocena z godz uspr: '); readln(klasa[i].Godziny_u);
        write('Ocena z godz nieuspr: '); readln(klasa[i].Godziny_n);
        write(BazaDanych,klasa[i]);
            {zapisanie w zmiennej zawartosci tablicy }
        writeln('czy chesz wpisowac nastepna osobe t/n');
        dalej:=readkey;
    Until (dalej='n')or(dalej='N');
    close(BazaDanych);
    menu;
end;

Procedure wyszukiwanie_nazwisko;
var i,niema:byte;
    szukane_nazwisko:string[20];
begin
    niema:=0; {zmienna pomocnicza 0-gdy nie znaleziono osoby}
    clrscr;
    assign(BazaDanych,'dziennik.dat');
    reset(BazaDanych);
    writeln('Podaj szukane nazwisko');
    read(szukane_nazwisko);
    Writeln('wcisnij dowolny klawisz');
    repeat until Keypressed;
    readln(dalej);
    i:=0;
    while not eof(BazaDanych) do {wykonuj az do konca pliku}
    begin
        i:=i+1;
        read(BazaDanych,klasa[i]);
        with klasa[i] do
            begin
                if szukane_nazwisko=klasa[i].nazwisko then
                    begin
                        niema:=1;
                        writeln;

```

```

        write('Imie: ');          writeln(klasa[i].imie);
        write('Nazwisko: ');      writeln(klasa[i].nazwisko);
        write('Oena z Jezyka Polskiego: '); writeln(klasa[i].O_polski);
        write('Ocena z matmy: ');  writeln(klasa[i].O_matma);
        write('Ocena z godz uspr: '); writeln(klasa[i].Godziny_u);
        write('Ocena z godz nieuspr: '); writeln(klasa[i].Godziny_n);
        writeln;
        writeln('wcisnij dowolny klawisz');
        dalej:=readkey;
    end;
end;
close(BazaDanych);
clrscr;
if niema=0 then
begin
    writeln('nie ma osoby o podanym nazwisku');
    writeln;
    writeln('wcisnij dowolny klawisz');
    readln;
end;
menu;
end;

procedure dopisywanie_danych; {procedura dopisywania danych}
var i:byte;
begin
    clrscr;
    assign(BazaDanych,'dziennik.dat');
    reset(BazaDanych);      {otwieram plik }
    i:=0;
    writeln('wcisnij dowolny klawisz');
    dalej:=readkey;
    Repeat
    clrscr;
    seek(BazaDanych,FileSize(BazaDanych));
    i:=i+1;
    writeln;
    Writeln('Podaj dane osoby do wczytania');
    write('Imie: ');          readln(klasa[i].imie);
    write('Nazwisko: ');      readln(klasa[i].nazwisko);
    write('Oena z Jezyka Polskiego: '); readln(klasa[i].O_polski);
    write('Ocena z matmy: ');  readln(klasa[i].O_matma);
    write('Ocena z godz uspr: '); readln(klasa[i].Godziny_u);
    write('Ocena z godz nieuspr: '); readln(klasa[i].Godziny_n);
    write(BazaDanych,klasa[i]);
        {zapisanie w zmiennej zawartosci tablicy }
    writeln('czy chesz wpisowac nastepna osobe t/n');
    dalej:=readkey;
    Until (dalej='n')or(dalej='N');
    close(BazaDanych);
    menu;
end;

procedure wyczytanie_danych; {procedura wyprowadzania danych}
var i:byte;
begin
    clrscr;
    Writeln('wcisnij dowolny klawisz');
    repeat until Keypressed;
    readln(dalej);
    assign(BazaDanych,'dziennik.dat');
    reset(BazaDanych);

```

```

i:=0;
while not eof(BazaDanych) do { wykonuj az do konca pliku}
begin
  i:=i+1;
  read(BazaDanych,klasa[i]);
  with klasa[i] do
    begin
      writeln;
      write('Imie: ');          writeln(klasa[i].imie);
      write('Nazwisko: ');      writeln(klasa[i].nazwisko);
      write('Oena z Języka Polskiego: '); writeln(klasa[i].O_polski);
      write('Ocena z matmy: ');  writeln(klasa[i].O_matma);
      write('Ocena z godz uspr: '); writeln(klasa[i].Godziny_u);
      write('Ocena z godz nieuspr: '); writeln(klasa[i].Godziny_n);
    end;
  end;
close(BazaDanych);
readln;
menu;
end;

```

```

Procedure wyswietl(ktory,kolor,tlo:byte);
Begin
  TextColor(kolor);
  TextBackGround(tlo);
  GotoXY(20,11+ktory);
  Write(tablica[ktory]);
End;

```

```

procedure menu;
begin
  Tablica[1]:='Pierwsze wprowadzanie danych-kasuje poprzednie';
  Tablica[2]:='      Wypisanie danych      ';
  Tablica[3]:='      Dopisywanie danych      ';
  Tablica[4]:='      Wyszukiwanie danych w/g nazwisk ';
  Tablica[5]:='      Wyjscie      ';
  ClrScr;
  For y:=1 To 5 Do
    wyswietl(y,15,0);
  y:=1;
  Repeat
    wyswietl(y,0,15);
    c1:=ReadKey;
    c2:=' ';
    If c1=chr(0) Then c2:=ReadKey;
    wyswietl(y,15,0);
    If c2='H' Then Dec(y);
    If c2='P' Then Inc(y);
    If y=0 Then y:=5;
    If y=6 Then y:=1;
    If c1=chr(27) Then y:=5;
  Until (c1=chr(27)) or (c1=chr(13));
  If y=1 Then wprowadz_dane;
  If y=2 Then wyczytanie_danych;
  If y=3 Then dopisywanie_danych;
  If y=4 Then wyszukiwanie_nazwisko;
end;

```

```

begin
  clrscr;
  menu;
end.

```


PLIKI TEKSTOWE

PRZYKŁAD 37

Program oblicza średnią arytmetyczną, geometryczną oraz harmoniczną trzech liczb zapisanych w pliku dyskowym. Wynik obliczeń zapisz w pliku dyskowym wraz z komentarzami. Program będzie sprawdzała istnienie pliku z danymi wejściowymi.

Specyfikacja danych:

Dane wejściowe: wartość zmiennych a b c (a=numer w dzienniku, b=miesiąc urodzenia, c=dzień urodzenia) zapisanych w plik z danymi o nazwie numer_w_dzienniku.in Plik ten wykonaj sam w dowolnym edytorze i zapisz na dysku a: czyli przed przystąpieniem do rozwiązywania zadania należy stworzyć plik w dowolnym edytorze tekstowym np. edytorze pascala lub notatniku z trzema liczbami oddzielonymi spacjami.

Dane wyjściowe wartości średniej arytmetycznej, geometrycznej, harmoniczej wraz z komentarzami. Wszystkie dane wyjściowe zapisane w pliku dyskowym o nazwie numer_w_dzienniku.out.

Uwagi odnośnie zaliczenia

Po wykonaniu zadania obejrzyj wynik obliczeń poprzez wyświetleniu pliku numer_w_dzienniku.out w dowolnym edytorze tekstowym, będziesz również uruchamiać program tak więc przygotuj listing programu w edytorze TP (poprawny bez błędów kompilacji oraz wykonania) teraz poproś nauczyciela w celu sprawdzenia.

Wzory:

<p>Średnia arytmetyczna</p> $s_a = \frac{\sum_{i=1}^n x_i}{n}$ $\sum_{i=1}^n x_i = x_1 + x_2 + \dots + x_n$ <p>symbol sumy Gdzie: x_i – wartość i-tego elementu n – ilość elementów</p>	<p>Średnia geometryczna</p> $s_g = \sqrt[n]{\prod_{i=1}^n x_i}$ <p>np.</p> $\prod_{i=1}^5 x_i = x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5$ <p>symbol iloczynu</p>	<p>Średnia harmoniczna</p> $s_h = \frac{1}{\sum_{i=1}^n \frac{1}{x_i}}$
---	--	--

{I-} dyrektywa dla kompilatora, wyłącza obsługę typu I/O nieprawidłowości operacji zapisu i odczytu z urządzeń zewnętrznych nie jest sygnalizowany.

{I+} dyrektywa dla kompilatora, włącza obsługę typu I/O nieprawidłowości operacji zapisu i odczytu z urządzeń zewnętrznych jest sygnalizowany.

IOResult wartość tej zmiennej przyjmuje zero jeśli podczas operacji I/O nie wystąpił błąd a inną wartość przy błędzie.

Uwaga: Konieczne będzie przypomnienie jak zapisujemy inaczej pierwiastek n-tego stopnia z liczby oraz jak go zapisujemy w Pascalu.

Do wykonania:

- Wpisz następujący program i sprawdź jego działanie
- Wpisz zawartość pliku wejściowego i wyjściowego do zeszytu.

Np.

Plik wejściowy

6 7 8

Plik wyjściowy

SREDNIA ARYTMETYCZNA

16.67

SREDNIA GEOMETRYCZNA

15.22

SREDNIA HARMONICZNA

4.54

```
PROGRAM ODCZYTANIE_DANYCH_Z_PLIKU_ZEWNETRZNEGO;
USES
```

```

    CRT;
VAR
    A,B,C:REAL;
    S_A,S_G,S_H:REAL;
    ZBIOR_Z_DANYMI,ZBIOR_Z_WYNIKAMI:TEXT;
BEGIN
    CLRSCR;
    ASSIGN(ZBIOR_Z_DANYMI,'A:WEJSCIE.DAT');
    {$I-}
    RESET(ZBIOR_Z_DANYMI);
    {$I+}
    IF IORESULT <> 0 THEN
        BEGIN
            WRITELN(' NIE MA PLIKU Z DANYMI WEJSCIOWYMI-PRZERYWAM PRACE');
            REPEAT UNTIL KEYPRESSED;
            HALT(0);
        END;
    READ(ZBIOR_Z_DANYMI,A,B,C);
    CLOSE(ZBIOR_Z_DANYMI);
    S_A:=(A+B+C)/3;
    S_G:=EXP(1/3*LN(ABS(A*B*C)));
    S_H:=1/(1/A+1/B+1/C);
    ASSIGN(ZBIOR_Z_WYNIKAMI,'A:WYJSCIE.DAT');
    REWRITE(ZBIOR_Z_WYNIKAMI);
    WRITELN(ZBIOR_Z_WYNIKAMI,'SREDNIA ARYTMETYCZNA');
    WRITELN(ZBIOR_Z_WYNIKAMI,S_A:6:2);
    WRITELN(ZBIOR_Z_WYNIKAMI,'SREDNIA GEOMETRYCZNA');
    WRITELN(ZBIOR_Z_WYNIKAMI,S_G:6:2);
    WRITELN(ZBIOR_Z_WYNIKAMI,'SREDNIA HARMONICZNA');
    WRITELN(ZBIOR_Z_WYNIKAMI,S_H:6:2);
    CLOSE(ZBIOR_Z_WYNIKAMI);
END.

```

ZADANIE 92.

Napisz program oblicza, energię kinetyczną oraz potencjalną po podaniu wysokości [m], masy[kg] oraz prędkości[km/h], wielkości te zostaną zapisane w pliku. Wynik obliczeń zapisz w pliku dyskowym wraz z komentarzami. Program będzie sprawdzała istnienie pliku z danymi wejściowymi.

Specyfikacja danych:

Dane wejściowe: wartość zmiennych h, m, v (m=numer w dzienniku, h=miesiąc urodzenia, v=dzień urodzenia) zapisanych w plik z danymi o nazwie ener_numer_w_dzienniku.in np. ener_45.in. Plik ten wykonaj sam w dowolnym edytorze czyli przed przystąpieniem do rozwiązywania zadania należy stworzyć plik w dowolnym edytorze tekstowym np. edytorze pascala lub notatniku z trzema liczbami oddzielonymi spacjami.

Dane wyjściowe wartości energii kinetycznej oraz potencjalnej wraz z komentarzami oraz jednostką
Wszystkie dane wyjściowe zapisane w pliku dyskowym o nazwie ener_numer_w_dzienniku.out ener_45.out

Wygląd pliku → energia_numer_w_dzienniku.out

Np.

Dane wejściowe:

h=10 m

m= 1 kg

v=72 km/h

Dane wyjściowe:

Energia kinetyczna

Ek=200 J

Energia potencjalna

Ep= 98.1 J

Uwaga:

W pliku wejściowym zapisana jest prędkość wyrażona w [km/h] do obliczeń używamy układu SI jednostek tak, więc konieczne będzie przeliczenie prędkości na [m/s].

Wzory:

$$E_k = \frac{m \cdot v^2}{2}$$

$$E_p = m \cdot g \cdot h$$

Ep-energia potencjalna [J]

Ek-energia kinetyczna [J]

g-przyspieszenie ziemskie [m/(s²)]

m-masa ciała [kg]

v-prędkość ciała [m/s]

h-wysokość, na której znajduje się ciało [m]

PRZYKŁAD 37a

Przykład wykonuje następujące czynności:

- sprawdza czy plik liczby.in znajduje się na dysku (pobierz plik od nauczyciela),
- wczytuje 500 liczb całkowitych z pliku tekstowego o nazwie liczby.in,
- znajduje element maksymalny i minimalny w ciągu A,
- zapisuje element maksymalny oraz minimalny na dysk do pliku o nazwie MAX_MIN.OUT,
- oblicza wartość ciągu B wg wzoru $B=A^2+1$,
- zapisuje ciąg B na dysk do pliku A_KWA_1.OUT.

Sprawdź w dowolnym edytorze pliki wynikowe.

```
PROGRAM MIN_MAX_Z_PLIKU;
USES
  CRT;
VAR
  MIN,MAX:INTEGER;
  A,B:ARRAY[1..500] OF INTEGER;
  I:INTEGER;
  ZBIOR_Z_DANYMI,ZBIOR_Z_WYNIKAMI:TEXT;
BEGIN
  CLRSCR;
  ASSIGN(ZBIOR_Z_DANYMI,'liczby.in');
  {$I-}
  RESET(ZBIOR_Z_DANYMI);
  {$I+}
  IF IORESULT<>0 THEN
    BEGIN
      WRITELN(' NIE MA PLIKU Z DANYMI WEJSCIOWYMI-PRZERYWAM PRACE');
      REPEAT UNTIL KEYPRESSED;
      HALT(0);
    END;

  { WCZYTANIE TABLICY A Z PLIKU }
  FOR I:=1 TO 500 DO
    BEGIN
      READ(ZBIOR_Z_DANYMI,A[I]);
    END;

  MAX:=A[1]; {tu jest jedynka}
  MIN:=A[1]; {tu jest jedynka}
  FOR I:=1 TO 500 DO
    BEGIN
      IF A[I]>MAX THEN MAX:=A[I];
      IF A[I]<MIN THEN MIN:=A[I];
    END;
  ASSIGN(ZBIOR_Z_WYNIKAMI,'MAX_MIN.OUT');
  REWRITE(ZBIOR_Z_WYNIKAMI);
  WRITELN(ZBIOR_Z_WYNIKAMI,MAX);
  WRITELN(ZBIOR_Z_WYNIKAMI,MIN);
  CLOSE(ZBIOR_Z_WYNIKAMI);

  ASSIGN(ZBIOR_Z_WYNIKAMI,'A_KWA_1.OUT');
```

```

REWRITE(ZBIOR_Z_WYNIKAMI);
FOR I:=1 TO 500 DO
BEGIN
  B[I]:=A[I]*A[I]+1;
  WRITELN(ZBIOR_Z_WYNIKAMI,B[I]);
END;
CLOSE(ZBIOR_Z_WYNIKAMI);
END.

```

ZADANIE 93.

Dokonaj tablicowania funkcji z krokiem numer w dzienniku/100 np. 0.25 określoną następująco:

$$f(x) = \begin{cases} \log_2 x & x \in (1;4 > \\ 1-x^2 & x \in <-2;1 > \end{cases}$$

Specyfikacja danych:

Dane wejściowe: brak danych wejściowych.

Dane wyjściowe: wartości argumentu oraz wartość funkcji dla tego argumentu. Wszystkie dane wyjściowe zapisane w pliku dyskowym o nazwie numer_w_dzienniku.out.
Formatowanie danych dwa miejsca po przecinku
Przykład zawartości pliku tekstowego:

Wynik tablicowania funkcji ucznia → tutaj wpisz nazwisko
f(-2.00)=-3.00

.....
.....

Uwagi odnośnie zaliczenia

Po wykonaniu zadania obejrzyj wynik obliczeń poprzez wyświetleniu pliku numer_w_dzienniku.out w dowolnym edytorze tekstowym, będziesz również uruchamiać program tak więc przygotuj listing programu w edytorze TP (poprawny bez błędów kompilacji oraz wykonania) teraz poproś nauczyciela w celu sprawdzenia.

Wzory:

W pascalu nie istnieje logarytm o dowolnej podstawie. W celu obliczenia logarytmu o dowolnej podstawie z użyciem logarytmu naturalnego LN zastosuj wzór:

$$\log_a b = \frac{LN(b)}{LN(a)}$$

ZADANIE 94.

Program do generowania losowej tablicy z przedziału <-6,8> o stu elementach a następnie obliczania średniej wylosowanych liczb.

Nazwa pliku z losowanymi liczbami to:

Cztery_pierwsze_litery_imienia_ucznia_los.in np. zoch_los.in

Dwa pierwsze wiersze tego pliku to:

Wylosowane liczby -- > Kowalski (tutaj Twoje nazwisko)

6

.....

Nazwa pliku ze średnią

Cztery_pierwsze_litery_imienia_ucznia_los.out np. zoch_los.out

Dwa pierwsze wiersze tego pliku to:

średnia -- > Kowalski (tutaj Twoje nazwisko)

3.23

ZADANIE 95.

Napisz program do generowania losowej tablicy z przedziału <0,255> o ilości elementów=numer w dzienniku + 200 a następnie zamiana tych liczb na system dwójkowy w jednym bajcie.

Nazwa pliku z losowanymi liczbami to:

Cztery_pierwsze_litery_nazwiska_ucznia_los.in np. kowa_los.in

Dwa pierwsze wiersze tego pliku to:

Wylosowane liczby -- > Kowalski (tutaj Twoje nazwisko)

145

.....

.....

Nazwa pliku z zamienionymi liczbami z systemu dziesiętnego na dwójkowy to:

Cztery_pierwsze_litery_nazwiska_ucznia_2.out np. kowa_2.out

Dwa pierwsze wiersze tego pliku to:

Liczby dziesiętne - dwójkowe --> Kowalski (tutaj Twoje nazwisko)

145 1001001

.....

.....

PRZYKŁAD 38

Program do mnożenia macierzy przez liczbę.

Specyfikacja danych:

Dane wejściowe: wartość zmiennej n (n =numer w dzienniku) i dowolna tablica 5x5 zapisana w pliku z danymi o nazwie numer_w_dzienniku.in Plik ten wykonaj sam w dowolnym edytorze i zapisz na dysku a: czyli przed przystąpieniem do rozwiązywania zadania należy stworzyć plik w dowolnym edytorze tekstowym np. edytorze pascala lub notatniku z trzema liczbami oddzielonymi spacjami.

np.

4 → liczba mnożąca tablicę

1 4 5 1 2 → macierz wejściowa

2 4 7 2 1

2 5 3 3 4

1 6 3 3 4

2 3 4 3 1

Dane wyjściowe

Dane wyjściowe zapisane w pliku dyskowym o nazwie numer_w_dzienniku.out.

4 16 20 4 8 → macierz wyjściowa (po mnożeniu)

8 16 28 8 4

8 20 12 12 16

4 24 12 12 16

6 12 16 12 4

Uwagi odnośnie zaliczenia

Po wykonaniu zadania obejrzyj wynik obliczeń poprzez wyświetlenie pliku numer_w_dzienniku.out w dowolnym edytorze tekstowym, będziesz również uruchamiać program tak więc przygotuj listing programu w edytorze TP (poprawny bez błędów kompilacji oraz wykonania) teraz poproś nauczyciela w celu sprawdzenia..

Wzory:

Mnożenie macierzy odbywa się według wzoru

$$b_{ij} = n \bullet a_{ij}$$

gdzie :

n – liczba

a_{ij} – macierz dana

b_{ij} – macierz wyjściowa

PROGRAM MNOZENIE_MACIERZY_PRZEZ_LICZBE_N;

USES

CRT;

VAR

A,B:ARRAY[1..5,1..5] OF REAL;

I,J:INTEGER;

N:REAL;

ZBIOR_Z_TABLICA:TEXT;

BEGIN

CLRSCR;

ASSIGN(ZBIOR_Z_TABLICA,'A:37.IN');

RESET(ZBIOR_Z_TABLICA);

{ WCZYTANIE ZMIENNEJ N Z PLIKU 37.IN }

READ(ZBIOR_Z_TABLICA,N);

{ WCZYTANIE TABLICY A Z PLIKU 37.IN }

```

FOR I:=1 TO 5 DO
  BEGIN
    FOR J:=1 TO 5 DO
      BEGIN
        READ(ZBIOR_Z_TABLICA,A[I,J]);
        END;
      READLN(ZBIOR_Z_TABLICA);
    END;
  FOR I:=1 TO 5 DO
    BEGIN
      FOR J:=1 TO 5 DO
        BEGIN
          B[I,J]:=N*A[I,J];
          END;
        END;
      ASSIGN(ZBIOR_Z_TABLICA,'A:37.OUT');
      REWRITE(ZBIOR_Z_TABLICA);
      WRITELN(ZBIOR_Z_TABLICA,N:6:2);
    { ZAPISANIE TABLICY A DO PLIKU 37.OUT}
    FOR I:=1 TO 5 DO
      BEGIN
        FOR J:=1 TO 5 DO
          BEGIN
            WRITE(ZBIOR_Z_TABLICA,A[I,J]:6:2,' ');
            END;
          WRITELN(ZBIOR_Z_TABLICA);
        END;
      { ZAPISANIE PUSTEGO WIERSZA DO PLIKU 37.OUT}
      WRITELN(ZBIOR_Z_TABLICA);
    { ZAPISANIE TABLICY B DO PLIKU 37.OUT}
    FOR I:=1 TO 5 DO
      BEGIN
        FOR J:=1 TO 5 DO
          BEGIN
            WRITE(ZBIOR_Z_TABLICA,B[I,J]:6:2,' ');
            END;
          WRITELN(ZBIOR_Z_TABLICA);
        END;
      CLOSE(ZBIOR_Z_TABLICA);
    END.

```

ZADANIE 96.

Napisz program wykonujący następujące czynności:

- zapytanie programu o wymiary macierzy (ilość wierszy oraz ilość kolumn),
- losowe generowanie wyrazów tablicy (macierzy) z przedziału $<-50,50>$,
- zapisanie wierszami wygenerowanej tablicy do pliku tekstowego i zapisanie na dyskiecie ucznia pod nazwą numer_z_dziennika_gen.out (ten sam plik tekstowy dla wszystkich poleceń),
- znalezienie maksymalnego oraz minimalnego wyrazu macierzy i zapisanie do pliku tekstowego,
- obliczenie i zapisanie macierzy do pliku tekstowego według wzoru:

$$b_{ij} = \left| \frac{a_{ij}}{a_{ij-\max} - a_{ij-\min}} \right|$$

gdzie :

a_{ij} – elementy macierzy losowanej

$a_{ij-\max}$ – element maksymalny

$a_{ij-\min}$ – element minimalny

- wykonaj specyfikację dla zadania i zapisz w zeszycie,

- oszacuj oraz zapisz w zeszycie złożoność obliczeń dla tego zadania czyli ile zostało wykonane losowań, porównań przy znajdowaniu elementu max i min, operacji arytmetycznych oraz zapisywań do plików tekstowych. W zeszycie zapisz ilość tych operacji wraz ze wzorem, z którego to wynika np. $8*7+2+8*7=114$ oraz krótkie słowne wytłumaczenie. Przed szacowaniem zapytaj nauczyciela ile generować wierszy i kolumn macierzy.

Uwagi odnośnie zaliczenia

- sprawdzenie specyfikacji zadania,
- po wykonaniu zadania obejrzyj wynik obliczeń poprzez wyświetlenie pliku numer_z_dziennika_gen.out w dowolnym edytorze tekstowym, będziesz również uruchamiać program tak więc przygotuj listing programu w edytorze TP (poprawny bez błędów kompilacji oraz wykonania) teraz poproś nauczyciela w celu sprawdzenia.

PRZYKŁAD 39

Zanotuj w zeszycie oraz naucz się znaczenie oraz zastosowanie instrukcji WITH (notatka łącznie z przykładem oraz wytłumaczeniem słownym).

Program jest rozwiązaniem zadania o następującej treści.

- wczytaj bazę danych w postaci pliku tekstowego o następujących polach: numer, nazwisko, imię,
- utwórz plik tekstowy zawierający osoby rozpoczynające się na literę B lub G,
- utworzenie bazy danych w postaci pliku tekstowego z kodami wszystkich pracowników. Kod powstanie poprzez pierwszą literę nazwiska pierwszą literę imienia oraz numer osoby,
- utwórz alfabetycznie uporządkowany plik tekstowy zawierający osoby, których imię jest Anna (pamiętaj o zdrobnieniach).

Specyfikacja danych:

Dane wejściowe:

Plik tekstowy o nazwie uczen.txt zapisany na dyskietce ucznia (poproś o ten plik od nauczyciela)

Dane wyjściowe

- plik tekstowy o nazwie numer_z_dzienika_dane.out zawierający osoby rozpoczynające się na literę B lub G,
- plik tekstowy o nazwie numer_z_dzienika_symbo.out zawierający kodami wszystkich pracowników,
- plik tekstowy o nazwie numer_z_dzienika_sort.out zawierający osoby, których imię jest Anna.

Uwagi odnośnie zaliczenia

- Sprawdzenie notatki (instrukcja WITH) oraz odpowiedź ustna.
- Specyfikacja problemu.**
- Po wykonaniu zadania obejrzyj wynik obliczeń poprzez wyświetlenie plików numer_z_dzienika_dane.out, numer_z_dzienika_symbo.out, numer_z_dzienika_sort.out dowolnym edytorze tekstowym, będziesz również uruchamiać program tak więc przygotuj listing programu w edytorze TP (poprawny bez błędów kompilacji oraz wykonania) teraz poproś nauczyciela w celu sprawdzenia.
- Zapisz w zeszycie nazwy trzech plików wynikowych oraz trzy pierwsze wiersze każdego pliku.**

uwagi:

- w przykładzie jedna pętla wykonywa jest z użyciem instrukcji WITH a reszta bez użycia tej instrukcji,
- dane w pliku uczen.txt nie są zapisane w równych kolumnach ponieważ nazwiska i imiona są różnej długości, dlatego aby wczytać dane z pliku tekstowego do pól bazy należy stosować metody z określaniem miejsca spacji w tekście jako znaku oddzielających dane z pól.

PROGRAM WCZYTYWANIA_PLIKU TEKSTOWEGO_DO_BAZY;

USES

CRT;

TYPE

BAZA=RECORD

NUMER: INTEGER;

IMIE: STRING[10];

NAZWISKO: STRING[20];

END;

VAR

DANE:ARRAY[1..1000] OF BAZA;

I,J,NUMER,SPACJA,ZAMIANA,NUMER_Z:INTEGER;

N:REAL;

IMIE_NAZWISKO,I_N,I_I,NN,NUMEREK,NAZWISKO_Z,IMIE_Z:STRING[40];

ZBIOR_Z_BAZA:TEXT;

```

BEGIN
CLRSCR;
ASSIGN(ZBIOR_Z_BAZA,' UCZEN.TXT');
RESET(ZBIOR_Z_BAZA);
{ WCZYTANIE DANYCH Z UCZEN.TXT DO BAZY }
I:=1;
WHILE NOT EOF(ZBIOR_Z_BAZA) DO
BEGIN
    READ(ZBIOR_Z_BAZA,NUMER);
    DANE[I].NUMER:=NUMER;
    READ(ZBIOR_Z_BAZA,IMIE_NAZWISKO);
{ WYCIECIE SPACJI Z PIERWSZEJ POZYCJI ZMIENNEJ IMIE_NAZWISKO }
    DELETE(IMIE_NAZWISKO,1,1);
{ OKRESLENIE MIEJSCA SPACJI ROZDZIELAJACEJ IMIE I NAZWISKO }
    SPACJA:=POS(' ',IMIE_NAZWISKO);
    DANE[I].IMIE:=COPY(IMIE_NAZWISKO,1,SPACJA-1);
    DANE[I].NAZWISKO:=COPY(IMIE_NAZWISKO,SPACJA+1,LENGTH(IMIE_NAZWISKO));
    I:=I+1; { LICZNIK REKORDOW W ZBIORZE WEJSCIOWYM }
    READLN(ZBIOR_Z_BAZA); { PRZEJSCIE DO NOWEGO REKORDU W ZBIORZE WEJSCIOWYM }
END;
CLOSE(ZBIOR_Z_BAZA);
ASSIGN(ZBIOR_Z_BAZA,' 37_DANE.OUT');
REWRITE(ZBIOR_Z_BAZA);
{ ZWRÓĆ UWAGĘ NA INSTRUKCJĘ WITH }
FOR J:=1 TO I-1 DO
    WITH DANE[J] DO
        BEGIN
            I_N:=NAZWISKO;
{ WPISANIE PIERWSZEJ LITERY Z POLA NAZWISKO DO ZMIENNEJ I_N }
            DELETE(I_N,2,LENGTH(I_N));
            IF (I_N='G') OR (I_N='B') THEN
                WRITELN(ZBIOR_Z_BAZA,NUMER,' ',IMIE,' ',NAZWISKO);
            END;
CLOSE(ZBIOR_Z_BAZA);
ASSIGN(ZBIOR_Z_BAZA,' 37_SYMBOL.OUT');
REWRITE(ZBIOR_Z_BAZA);
FOR J:=1 TO I-1 DO
    BEGIN
        I_N:=DANE[J].NAZWISKO;
        I_I:=DANE[J].IMIE;
{ WPISANIE PIERWSZEJ LITERY Z POLA IMIE DO ZMIENNEJ I_I }
        DELETE(I_N,2,LENGTH(I_N));
        DELETE(I_I,2,LENGTH(I_I));
        STR(DANE[J].NUMER,NUMEREK);
        WRITELN(ZBIOR_Z_BAZA,I_N+I_I+NUMEREK);
        END;
CLOSE(ZBIOR_Z_BAZA);
{ ALFABETYCZNE SORTOWANIE CAŁEJ BAZY }
REPEAT
    ZAMIANA:=0;
    FOR J:=1 TO I-2 DO
        BEGIN
            IF DANE[J].NAZWISKO>DANE[J+1].NAZWISKO THEN
                BEGIN
                    NUMER_Z:=DANE[J].NUMER;
                    NAZWISKO_Z:=DANE[J].NAZWISKO;
                    IMIE_Z:=DANE[J].IMIE;
                    DANE[J].NUMER:=DANE[J+1].NUMER;
                    DANE[J].NAZWISKO:=DANE[J+1].NAZWISKO;
                    DANE[J].IMIE:=DANE[J+1].IMIE;
                    DANE[J+1].NUMER:=NUMER_Z;
                    DANE[J+1].NAZWISKO:=NAZWISKO_Z;
                    DANE[J+1].IMIE:=IMIE_Z;
                END
            END
        END
    UNTIL ZAMIANA=0;
END

```



```

        ZAMIANA:=1;
    END;
END;
UNTIL ZAMIANA=0;
ASSIGN(ZBIOR_Z_BAZA,' 37_SORT.OUT');
REWRITE(ZBIOR_Z_BAZA);
FOR J:=1 TO I-1 DO
    BEGIN
        IF (DANE[J].IMIE='Anna') OR (DANE[J].IMIE='Ania') THEN
            WRITELN(ZBIOR_Z_BAZA,DANE[J].NUMER,' ',DANE[J].IMIE,' ',DANE[J].NAZWISKO);
        END;
    CLOSE(ZBIOR_Z_BAZA);
READLN;
END.

```

ZADANIE 97.

W pliku firma.txt (poproś nauczyciela o ten plik), znajdują się dane osób zatrudnionych w pewnej firmie. Dane jednej osoby są umieszczone w osobnym wierszu i zawierają: nazwisko, imię, datę urodzenia (dd-mm-rr), miejsce urodzenia, stanowisko zajmowane w firmie. Dane w wierszach są rozdzielone spacjami w taki sposób, że wszystkie dane tego samego typu rozpoczynają się w tej samej kolumnie. Przykład:

Kowal	Michał	02-12-69	Warszawa	sekretarka
Ciosek	Anna	22-08-64	Kraków	informatyk

a) Utwórz zestawienie, które zawiera wiersze z danymi osób z pliku firma.txt urodzonych w miejscowościach, których nazwa zaczyna się na pierwszą literę twojego nazwiska lub pierwszą literę Twojego imienia. Wynik zapisz na dyskietce ucznia pod nazwą numer_z_dziennika_zest.out

b) Utwórz zestawienie danych wszystkich pracowników firmy z ich kodami.

Kod pracownika składa się z ciągu następujących znaków: pierwszej litery nazwiska, pierwszej litery imienia oraz dwóch ostatnich cyfr z roku urodzenia pracownika. Litery występujące w kodzie pracownika mają być małe.

W zestawieniu dla każdego pracownika, w osobnym wierszu zamieść jego następujące dane: imię, nazwisko, data urodzenia, kod. Postać wiersza zestawienia odczytaj z poniższego przykładu:

Jan Nowak 12-05-69 nj69

zestawienia oraz umieść 18 pierwszych wierszy tego zestawienia. Wynik zapisz na dyskietce ucznia pod nazwą numer_z_dziennika_kod.out

c) Utwórz zestawienie osób zatrudnionych w firmie na stanowisku grafik, uporządkowane alfabetycznie ze względu na nazwisko. W zestawieniu dla każdego pracownika, w osobnym wierszu, zamieść jego następujące dane: imię nazwisko. Postać wiersza zestawienia odczytaj z poniższego przykładu:

Jan Nowak

Wynik zapisz na dyskietce ucznia pod nazwą numer_z_dziennika_graf.out

Uwaga:

Gdy dane w pliku tekstowym zapisane są w równych kolumnach (tak jak w pliku firma.txt) to nie ma konieczności wykonywania operacji ze spacjami aby odczytać początek danych do nowego pola (patrz poprzedni przykład).

Wystarczy użyć w odpowiedniej pętli np. READ(BAZA_DANE,BAZA[I].NAZWISKO,.....); Trzeba jednak zadeklarować strukturę bazy danych (rekordu) w TYPE w taki sposób aby szerokość kolumn danych odpowiadała długości pól rekordów (należy przeglądnąć plik firma.txt aby ustalić szerokość kolumn).

Uwagi odnośnie zaliczenia

Po wykonaniu zadania obejrzyj wynik obliczeń poprzez wyświetleniu plików numer_z_dzienika_zest.out, numer_z_dzienika_kod.out, numer_z_dzienika_graf.out dowolnym edytorze tekstowym, będziesz również uruchamiać program tak więc przygotuj listing programu w edytorze TP (poprawny bez błędów kompilacji oraz wykonania) teraz poproś nauczyciela w celu sprawdzenia.

ZBIORY

Wprowadzenie

Język Pascal umożliwia wykorzystywanie w programach zbiorów, których elementy muszą należeć do pewnego określonego typu.

Definiowanie.

Typ zbiorowy definiujemy w sposób następujący:

type *nazwa* = set of *nazwa_typu* ;

gdzie:

nazwa jest nazwą typu zbiorowego,

nazwa_typu jest nazwą tzw. typu podstawowego, który stanowi elementy zbioru (może to być typ integer, char, boolean, wyliczeniowy lub okrojony). Wartościami zmiennych typu zbiorowego są zatem wszystkie zbiory, które można utworzyć z wartości typu podstawowego oraz zbiór pusty (pusty nie zawiera żadnego elementu).

Wartości zmiennym typu zbiorowego nadajemy przy pomocy tzw. konstruktora zbioru, którym jest para nawiasów prostokątnych [], wewnątrz których wpisujemy wartości typu podstawowego.

Operacje na zbiorach

Język Pascal umożliwia wykonywanie następujących operacji na zbiorach:

Operator	Znaczenie
+	suma zbiorów
−	różnica zbiorów
*	część wspólna zbiorów

Należy również zwrócić uwagę na operator **in**, który umożliwia stwierdzenie, czy dany element należy do zbioru.

Można też wykorzystywać operatory relacyjne:

Operator	Znaczenie
=	równość zbiorów
<>	nierówność zbiorów
<= oraz >=	zawieranie się zbiorów

PRZYKŁAD 40

Program jest rozwiązaniem zadania o następującej treści.

➤ zdefiniowanie trzech zbiorów

$ZBIOR_A = \{A, B, C, D\}$

$ZBIOR_B = \{C, D, E, F\}$

$ZBIOR_C = \{E, F, G, H\}$

➤ określenie wyniku działania na zbiorach

$WYNIK = (ZBIOR_A \setminus ZBIOR_B) \cup ZBIOR_C$

➤ wyprowadzenie na ekran wyniku działania zdefiniowanego powyżej według algorytmu:

dla zmiennej I przebiegaj wszystkie wartości typu wyliczeniowego **wykonaj**

jeśli I zawiera się w wynik **to**

wyprowadź z użyciem instrukcji case odpowiedni tekst w zależności od wartości I

➤ rozstrzygnij czy ZBIOR_C zawiera się w WYNIKU

Polecenia do wykonania:

- wykonaj notatki → wprowadzenie, definiowanie, wartości zmiennym, operacje na zbiorach (dwie tabelki), znaczenie **in**,
- naucz się wykonanych notatek,
- wpisz treść przykładu i zapisz na dyskiecie ucznia po nazwę numer_z_dziennik.pas,
- uruchom program.

Specyfikacja danych:

Dane wejściowe:

- trzy zdefiniowane zbiory

Dane wyjściowe

- zbiór wynikowy zdefiniowany według działania określonego w treści zadania
- tekst określający zawieranie lub nie zbiorów

Uwagi odnośnie zaliczenia

- sprawdzenie notatek,
- odpowiedź ustana,
- uruchomienie oraz wykonanie programu.

PROGRAM DZIAŁANIA_NA_ZBIORACH;

USES

CRT;

TYPE

LITERY=(A,B,C,D,E,F,G,H);

Z_LITER=SET OF LITERY;

VAR

ZBIOR_A,ZBIOR_B,ZBIOR_C,WYNIK:Z_LITER;

```

I:LITERY;
BEGIN
CLRSCR;
ZBIOR_A:=[A,B,C,D];
ZBIOR_B:=[C,D,E,F];
ZBIOR_C:=[E,F,G,H];
WRITELN('Od zbioru A odjęłem B i do wyniku dodałem zbior C');
WRITELN('Elementy wyświetle poniżej');

WYNIK:=(ZBIOR_A-ZBIOR_B)+ZBIOR_C;
FOR I:=A TO H DO
  IF I IN WYNIK THEN
    CASE I OF
      A:WRITELN('A ');
      B:WRITELN('B ');
      C:WRITELN('C ');
      D:WRITELN('D ');
      E:WRITELN('E ');
      F:WRITELN('F ');
      G:WRITELN('G ');
      H:WRITELN('H ');
    END;
  WRITELN;
IF ZBIOR_C<=WYNIK THEN
  WRITELN('ZBIOR C ZAWIERA SIE W ZBIORZE WYNIK')
ELSE
  WRITELN('ZBIOR C NIE ZAWIERA SIE W ZBIORZE WYNIK');
REPEAT UNTIL KEYPRESSED;
END.

```

ZADANIE 98.

- Utwórz trzy zbiory towarów i zapisz je w zeszycie:

zbiór wszystkich towarów dostępnych na rynku → 10 towarów 3 towary na literę taka jaką jest Twoje nazwisko, 3 towary na literę taką jaką jest Twoje imię, 2 towary na drugą literę Twojego nazwiska, 2 towary na drugą literę Twojego imienia,

zbiór towarów dostępnych w sklepie A → 7 towarów 2 towary na literę taka jaką jest Twoje nazwisko, 2 towary na literę taką jaką jest Twoje imię, 2 towary na drugą literę Twojego nazwiska, 1 towar na drugą literę Twojego imienia,

zbiór towarów dostępnych w sklepie B → 7 towarów 2 towary na literę taka jaką jest Twoje nazwisko, 2 towary na literę taką jaką jest Twoje imię, 1 towar na drugą literę Twojego nazwiska, 2 towary na drugą literę Twojego imienia,

uwaga:

towary ze sklepu A i B muszą się zawierać w zbiorze wszystkich towarów oraz dwa towary muszą się powtórzyć w sklepie A i B

Wykonaj:

- zapisz w zeszycie symbolicznie działania na zbiorach używając zbiorów sklep A, sklep B, wszystkie towary oraz notacji używanej w matematyce czyli suma, różnica oraz iloczyn zbiorów dla stwierdzeń od 1) do 7) patrz poniżej,
 - zapisz w zeszycie specyfikację problemu,
 - napisz program definiujący trzy zbiory (takie jakie masz w już w zeszycie) oraz wykonujący następujące działania na zbiorach dla stwierdzeń od 1) do 7) patrz poniżej, wypisz wyniki tych działań wraz z komentarzem jakie działanie zostało wykonane:
- 1) artykuły dostępne w jednym lub drugim sklepie
 - 2) artykuły dostępne w obu sklepach
 - 3) artykuły dostępne w sklepie A i niedostępne w sklepie B
 - 4) artykuły dostępny w sklepie B i niedostępny w sklepie A
 - 5) artykuły niedostępny w sklepie A
 - 6) artykuły niedostępne w sklepie B
 - 7) artykuły niedostępne w żadnym sklepie

Uwagi odnośnie zaliczenia

- sprawdzenie notatek,
- odpowiedź ustana,
- uruchomienie oraz wykonanie programu.

Ciekawe programy → pewne pomysły:

–Zawieszanie sesji DOS (komputera):

```
REPEAT UNTIL FALSE;
```

–Reset komputera:

Instrukcja INLINE

Służy ona do dołączania w programie lub module paskalowym krótkich podprogramów lub instrukcji napisanych w kodzie maszynowym. Jej postać jest następująca:

INLINE (lista_elementów_inline);

przy czym poszczególne kody w postaci liczb szesnastkowych lub identyfikatorów, są rozdzielane znakiem „/”

Przykład:

```
INLINE ($CD/$12/$89/$46/$04);
```

Przykładowy program

```
PROCEDURE RESET;
```

```
  BEGIN
```

```
    INLINE ( $EA/ $F0/ $FF/ $00/ $F0) ;
```

```
  END;
```

–Wykonanie instrukcji DOSu przez program PASCALA np.

(wyświetlenie wszystkich zbiorów z rozszerzeniem EXE)

Instrukcja EXEC

EXEC(Sciezka do proga, parametry)

parametry mogą być:

0 - zakończenie normalne

1 - Ctrl c

2 - błąd urządzenia

```
PROGRAM WYS ;
```

```
  USES CRT , DOS ;
```

```
  {$M $4000 , 0, 0 }
```

```
  BEGIN
```

```
    CLRSCR;
```

```
    EXEC(' \COMMAND.COM ' , ' / C DIR * . EXE ' ) ; {dla DOS}
```

```
    EXEC('C:\WINNT\SYSTEM32\CMD.EXE ' , ' / C DIR * . EXE ' ) ; {dla WinNT4.0}
```

```
    REPEAT UNTIL KEYPRESSED;
```

```
  END.
```

–Drukowanie zbiorów tekstowych.

W celu wydrukowania zbiorów tekstowych lub zawartości zmiennych tekstowych należy w instrukcji WRITE(ZNAK) wpisać LST w następujący sposób WRITE(LST,ZNAK) gdy chcesz wydrukować linię pustą na drukarce to musisz wykonać instrukcję WRITELN(LST).

Uwaga: Konieczne dołączenie modułu PRINTER.

ZADANIE 99.

Na dysku C założyć folder nazwisko ucznia (bez polskich liter), tym folderze założyć folder imię ucznia (bez polskich liter). Do folderu imię ucznia dokonać kopiowania trzech dowolnych plików z rozszerzeniem PAS. Napisz program, który dokona wyświetlenia tych plików z folderu imię ucznia.

ZADANIE 100.

Napisać program i dokonać kompilacji na EXE. Nazwa programu nazwisko ucznia (bez polskich liter, do ośmiu znaków). Program powinien zawierać pięć opcji:

1) drukowanie nazwiska ucznia

2) pusta linia na drukarce

3) zawieszanie komputera

4) reset komputera

5)koniec programu

W celu zaliczenia przepis listning oraz poproś nauczyciela w celu sprawdzenia (czy nazwa poprawna, czy ma cztery opcje).

Zadania przykładowe na sprawdzian

ZADANIE 1

Napisz program, który po wczytaniu tekstu do zmiennej drukował będzie ten tekst na ekranie litera po literze w taki sposób, że każda litera tekstu pisana jest w kolumnie o jeden większej i w wierszu o jeden większym. Gdy skończą się kolumny ekranu to następuje zmniejszanie kolumn itd.

ZADANIE 2

Napisz program, który będzie sprawdzał następujące równości działania na zbiorach zdefiniowanych następująco:

$$A = \{1, 2, 3, 4\}$$

$$B = \{3, 4, 5, 6, 7\}$$

$$C = \{4, 5, 8, 9\}$$

sprawdzana równość to:

$$A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C)$$

W programie wykorzystaj definiowanie zbiorów oraz działania na nich

Napisz program w taki sposób aby program wyświetlał wynik prawej i lewej strony równości oraz program pisał jeden z napisów „tak równość zachodzi” lub „nie równość nie zachodzi”.

ZADANIE 3

Napisz program znajdowania wartości kwoty twojego kapitału złożonego w banku.

Dane:

trzy liczby zapisane w pliku tekstowym na dyskiecie ucznia o nazwie numer_z_dziennika_kap.in

- liczba pierwsza to kapitał,
- liczba druga to stopa procentowa jednego okresu oszczędzania,
- liczba trzecia to ilość okresów oszczędzania.

Wynik:

kapitał łączny wraz odsetkami po okresie oszczędzania zapisany w pliku tekstowym na dyskiecie ucznia o nazwie numer_z_dziennika_kap.out.

$$K_n = K \cdot \left(1 + \frac{p}{100}\right)^n$$

gdzie:

K_n – wkład czyli odsetki oraz kapitał

p – stopa procentowa

K – kapitał początkowy

n – ilość okresów oszczędzania

ZADANIE 4

W pliku firma.txt (poproś nauczyciela o ten plik), znajdują się dane osób zatrudnionych w pewnej firmie. Dane jednej osoby są umieszczone w osobnym wierszu i zawierają: nazwisko, imię, datę urodzenia (dd-mm-rr), miejsce urodzenia, stanowisko zajmowane w firmie. Dane w wierszach są rozdzielone spacjami w taki sposób, że wszystkie dane tego samego typu rozpoczynają się w tej samej kolumnie. Przykład:

Kowal	Michał	02-12-69	Warszawa	sekretarka
Ciosek	Anna	22-08-64	Kraków	informatyk

a) Utwórz zestawienie, które zawiera wiersze z danymi osób z pliku firma.txt, których nazwiska zaczynają się na T lub K. Wynik zapisz na dyskiecie ucznia pod nazwą numer_z_dziennika_zada.out

b) Utwórz zestawienie danych wszystkich pracowników firmy z rokiem urodzenia (bez dnia i miesiąca).

W zestawieniu dla każdego pracownika, w osobnym wierszu zamieść jego następujące dane: imię, nazwisko, rok urodzenia, kod. Postać wiersza zestawienia odczytaj z poniższego przykładu:

Jan Nowak 69

Wynik zapisz na dyskiecie ucznia pod nazwą numer_z_dziennika_zadb.out

c) Utwórz zestawienie osób zatrudnionych w firmie na stanowisku informatyk, uporządkowane alfabetycznie ze względu na imię od Z do A. W zestawieniu dla każdego pracownika, w osobnym wierszu, zamieść jego następujące dane: imię nazwisko stanowisko.

Wynik zapisz na dyskiecie ucznia pod nazwą numer_z_dziennika_zadc.out

ZADANIE 5

Napisz program, który w dowolnym pliku tekstowym, którego nazwa wczytywana jest z klawiatury zastąpi jedną spację dwiema. Plik ten zostanie zapisany na dyskiecie ucznia po nazwą numer_z_dziennika_zad5.out.

ZADANIE 6

Napisz program do transponowania macierzy.

Specyfikacja danych:

Dane wejściowe:

- dowolna macierz zapisana w pliku tekstowym na dyskiecie ucznia o nazwie numer_z_dziennika_maci.in o wymiarze 4×7

Dane wyjściowe

- macierz transponowana zapisana w pliku tekstowym na dyskiecie ucznia o nazwie numer_z_dziennika_tran.out

ZADANIE 7

Napisz program do podnoszenia macierzy do dowolnej potęgi naturalnej.

Specyfikacja danych:

Dane wejściowe:

- dowolna macierz zapisana w pliku tekstowym na dyskiecie ucznia o nazwie numer_z_dziennika_przed.in o wymiarze 4×4 oraz liczba naturalna \rightarrow potęga do, której podnosimy macierz

Dane wyjściowe

- macierz podniesiona do potęgi zapisana w pliku tekstowym na dyskiecie ucznia o nazwie numer_z_dziennika_pop.out

CZEŚĆ DZIEWIĄTA WYKŁADU Z PASCALA

Używanie Assemblera w Pascalu

Assembler jest językiem niskiego poziomu który służy do programowania wszelkiego rodzaju procesorów.

Kod assemblera jest wpisywany przez użytkownika i wyświetlany na ekranie w postaci **mnemoników**. Są to krótkie i łatwe do zapamiętania instrukcje. Do programowania nie potrzebna jest znajomość kodu maszynowego każdego procesora. Większość języków programowania jest tłumaczonych przez ich kompilatory na assembler. Program narzędziowy zwany assemblerem zamienia mnemoniki na kod maszynowy interpretowany przez procesor.

Rejestr procesora to pamięć podręczna procesora, składa się z niewielu komórek pamięci o pewnych nazwach np. AX.

Pisząc program w Pascalu stosujemy wstawki w assemblerze. Wstawka zaczyna się od słowa **asm** a kończy się **end;**

Przykład 41

Temat: Wstawianie wstawki assemblera. Nadanie zmiennej wynik wartość 10.

1)Zapisz w zeszycie co to jest:

- assembler
- mnemonik
- jak rozpoczyna i kończy się wstawka w assemblerze
- wytłumacz znaczenie mnemonika mov
- wytłumacz znaczenie mnemonika mul (znajdź w necie)
- co jest rejestr procesor. Podaj przykładową nazwę.

Wykonaj następująco w zeszycie (patrz poniżej):

assembler (to przepis a pod napisz co to jest)

.....

mnemonik (to przepis a pod napisz co to jest)

.....

2)Wpisz i przetestuj przykład.

3)Zapisz temat przykładu a poniżej jego treść wraz komentarzami.

```
Program przyklad_assembler1;
{$ASMMODE INTEL}    {tylko dla FP, określenie procesora INTEL}
uses
  crt;
var
  wynik:integer;
begin
  clrscr;
  writeln('Wynik umieszczony w rejestrze AX: ');
  asm
    mov ax, 10          {poczatek wstawki assemblera}
                        {przeniesienie do rejestru AX liczby 10}

    mov wynik , ax      {przeniesienie do zmiennej wynik wartosci }
                        {rejestru AX}
  end;                  {koniec wstawki assemblera}
  writeln('wynik=',wynik);
  repeat until keypressed;
end.
```

Przykład 42

Temat: Wstawianie wstawki assemblera. Mnożenie liczb 2 i 3.

1)Zapisz temat przykładu a poniżej jego treść wraz komentarzami.

```
Program przyklad_assembler2;
```



```

{$ASMMODE INTEL}    {tylko dla FP, określenie procesora INTEL}
uses
  crt;
var
  wynik:integer;
function mnozenie(x,y:byte):integer;
var
  m:integer;
begin
  asm
    mov al, x      {początek wstawki asemblera}
    mul y          {przeniesienie do rejestru al wartosci x}
    mov m, ax      {mnozenie przez wartosc y}
  end;             {przeniesienie do zmiennej m wartosci rejestru ax }
  mnozenie:=m;     {koniec wstawki asemblera}
end;

begin
  clrscr;
  writeln('mnozenie 2*3 =', mnozenie(2,3));
  repeat until keypressed;
end.

```

Sprawdzanie pamięci.

funkcja MemAvail →zwracająca sumaryczny rozmiar wolnej pamięci,
funkcja MaxAvail →zwracająca rozmiar największego wolnego bloku).

Przykład 43

Temat: Sprawdzanie ilości wolnej pamięci.

Wykonaj:

- 1)Uruchom przykład
- 2)Przepisz do zeszytu znaczenie instrukcji funkcja MemAvail, MaxAvail,
- 3)Dopisz do przykładu linię, która wyświetli największy wolny bloku pamięci
- 4)Przepisz przykład z Twoją poprawką

```

program IlePamieci;
  Uses Crt;
begin
  writeln('Masz w tej chwili ', MemAvail, ' bajtow wolnej pamieci. ');
  Repeat until keypressed;
end.

```

Dynamiczne struktury danych.

Przykład 44

Temat: Demonstracja możliwości obsługi zmiennych dynamicznych.

Wykonaj:

1) Udziel odpowiedzi na następujące pytania. Najpierw przepisuj pytanie pod nim odpowiedź.

Co to są zmienne globalne?

Co to są zmienne lokalne?

Gdzie są umieszczane zmienne globalne oraz jak jest ich wielkość.

Gdzie są umieszczane zmienne lokalne oraz jak jest ich wielkość.

Co to jest stos? Do czego można porównać.

Co to jest sarta?

Jaki jest okres życia zmiennych lokalnych i globalnych.

W jaki sposób można zwiększyć pamięć przeznaczoną na zmienne?

Opisz definicję wskaźnika do zmiennej.

Deklarację wskaźnika do zmiennej wraz z przykładem.

Odwołanie się do wskaźnika, wraz z przykładem

Narysuj schemat sposobu obsługi zmiennych statycznych i wskazywanych wraz z opisem. Opis odwołania do zmiennej wskaźnikowej.

Wymień powód stosowania wskaźników.

Co to są wskaźniki amorficzne?

2) Przepisz temat przykładu.

3) Uruchom program przykładu i przepisuj do zeszytu.

Program wskaźniki;

```
uses
```

```
  crt;
```

```
var
```

```
  wskaźnik: ^integer;
```

```
begin
```

```
  clrscr;
```

```
  new(wskaźnik);
```

```
  wskaźnik^:=4;
```

```
  writeln(wskaźnik^);
```

```
  dispose(wskaźnik);
```

```
  writeln(wskaźnik^);
```

```
  repeat until keypressed;
```

```
end.
```

Sposób przechowywania w pamięci zmiennych globalnych i lokalnych.

Zmienne, którymi posługiwałeś się do tej pory, należały do klasy zmiennych globalnych (jeśli zdefiniowałeś je na początku programu) lub lokalnych (jeżeli zostały zdefiniowane w procedurach).

Zmienne globalne umieszczane są w tzw. **segmencie danych**, którego wielkość wynosi 64 kB i jest stała.

Zmienne lokalne, przechowywane są w segmencie **stosu** (obszar pamięci przeznaczony do przechowywania danych lokalnych [wartości zmiennych lokalnych oraz argumenty procedur oraz wartości funkcji], elementu stosu są poukładane jeden na drugim, można ustalić który element jest pierwszy, oraz kolejność elementów przypomina to wieżę z klocków) którego wielkość można zmieniać w granicach od 1 do 64 kB.

Sarta Obszar pamięci przeznaczony na tzw. zmienne dynamiczne, tworzone i niszczone w zależności od potrzeb.

Długość życia zmiennych globalnych i lokalnych.

Zmienne globalne są statyczne, tj. istnieją przez cały czas wykonywania programu.

Zmienne lokalne "żyją" tylko tak długo, jak długo wykonują się posiadające je procedury lub funkcje. Koniec końców, na zmienne możemy przeznaczyć co najwyżej 128 kilobajtów, z czego 64 kB dostępne jest "warunkowo".

W jaki sposób można zwiększyć pamięć przeznaczoną na zmienne.

Sposobem na zwiększenie ilości pamięci dla zmiennych jest użycie **zmiennych wskazywanych**. W odróżnieniu od "zwykłych" zmiennych, zmienne wskazywane mogą być umieszczane w dowolnym miejscu pamięci, z czym wiąże się nieco inny sposób odwoływania do nich, wykorzystujący tzw. **wskaźniki**.

Opis wskaźnika.

Def:

Wskaźnik do zmiennej jest po prostu jej adresem, czyli liczbą opisującą jej położenie w pamięci.

Deklaracja wskaźnika:

Deklaracja wskaźnika do zmiennej określonego typu jest bardzo podobna do "zwykłej" deklaracji zmiennej i różni się od niej jedynie symbolem wskaźnika (^):

zmienna : ^typ np. droga:^integer;

Uwaga:

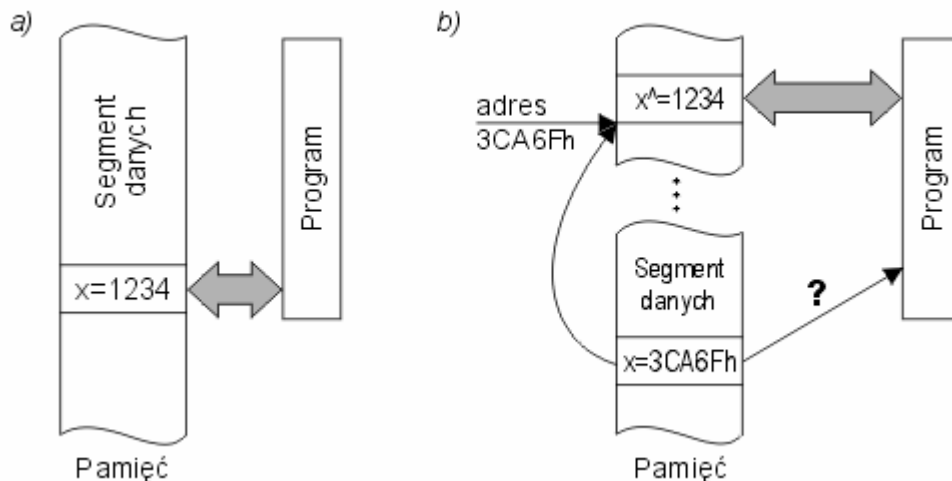
przy czym typ musi być identyfikatorem typu prostego lub zdefiniowanego wcześniej w sekcji type.

Odwołanie się do wskaźnika:

Odwołanie do samego wskaźnika wygląda dokładnie tak samo, jak dla każdej innej zmiennej, natomiast odwołanie do wskazywanej przez niego zmiennej zawiera dodatkowo znaczek ^, tym razem położony za nazwą wskaźnika:

np. writeln(zmienna^);

Sposób obsługi zmiennych statycznych (a) i wskazywanych (b).



Źródło: Turbo Pascal. Programowanie Autor: Tomasz M. Sadowski Copyright © 1996 by Wydawnictwo Helion.

Opis odwołania do zmiennej wskaźnikowej.

Jak widać, dostęp do zmiennej wskazywanej odbywa się dwuetapowo, za pośrednictwem wskaźnika: program odczytuje wartość tego ostatniego (czyli adres zmiennej) i na jej podstawie odwołuje się do właściwej zmiennej, położonej w pamięci w miejscu opisanym adresem (w naszym przypadku 3CA6Fh szesnastkowo). Metoda ta powoduje nieznaczne wydłużenie czasu dostępu do zmiennej, ale pozwala za to umieścić ją praktycznie w dowolnym miejscu pamięci (dokładnie, zmienne wskazywane umieszczane są na tzw. stercie (ang. heap) - wydzielonym obszarze pamięci zarządzanym przez specjalne procedury).

Tablicą wskaźników

```

type
    Tablica = array[1..100] of real; { tablica 100 liczb real }
    WskTab = ^Tablica; { wskaźnik do tablicy }
var
    Tablica_wsk: array[1..100] of WskTab; { tablica wskaźników }

{ ... }

Tablica_wsk[10]^ := 3.1415;
```

Tablica Tablica_wsk, będąca tablicą wskaźników do tablic liczb rzeczywistych. Tablica składającej się z 100 wierszy (wskazywanych wskaźnikami) zawierających po 100 liczb każdy, czyli w sumie 20 tysięcy liczb o objętości około 120 kB! Umieszczoną nieco niżej instrukcję `Tablica_wsk[10]^ := 3.1415` można wytłumaczyć jako odwołanie do dziesiątej komórki, czyli tablicy wskazywanej przez dziesiąty wskaźnik w tablicy Tablica_wsk.

Powód stosowania wskaźników.

- Głównym zastosowaniem wskaźników jest obsługa dużych struktur danych np. tablic rekordów.
- Drugą bardzo ważną i pożyteczną cechą zmiennych wskazywanych jest możliwość ich tworzenia i niszczenia w zależności od potrzeb.

Wskaźniki amorficzne

Oprócz wskaźników do obiektów konkretnego typu Turbo Pascal udostępnia również wskaźniki amorficzne, tj. wskazujące na obszar pamięci i nie związane z żadnym konkretnym typem. Wskaźnik taki deklarowany jest słowem **pointer (wskaźnik)** i jest zgodny pod względem przypisania z innymi typami wskaźnikowymi. Typ pointer i związane z nim operacje wykorzystywane są przez bardziej zaawansowanych programistów do bezpośredniego, "niskopoziomowego" manipulowania zawartością pamięci.

Instrukcje związane ze wskaźnikami.

new(wskaźnik-do-zmiennej)

Procedura new wykonuje czynności związane z utworzeniem zmiennej wskazywanej

dispose(wskaźnik-do-zmiennej)

Procedura dispose - operacje związane z jej usunięciem.

GetMem(wskaźnik, rozmiar-bloku)

FreeMem(wskaźnik, rozmiar-bloku)

Procedury te wykorzystują wskaźniki amorficzne (typu pointer) i służą do bardziej "wewnętrznego" manipulowania pamięcią. Wielkość przydzielanego lub zwalnianego bloku (w bajtach) określa parametr rozmiar-bloku.

mark(wskaźnik)

Wykonanie procedury mark nie powoduje przydzielenia pamięci ani utworzenia zmiennej, a jedynie zapamiętanie bieżącej "wysokości" sterty w zmiennej wskaźnik.

release(wskaźnik)

Zwolnienia całego obszaru sterty leżącego powyżej wskaźnika dokonuje się za pomocą procedury release.

Przykład 45

Temt: Demonstracja możliwości obsługi zmiennych dynamicznych.

- 1)Przepisz temat przykładu.
- 2)Wpisz i uruchom progrm.
- 3)Przepisz program wraz z komentarzami.
- 4)Zapisz w zeszycie pięć uwag, które są umieszczone pod listingiem programu.

```
program ZmienneDynamiczne;
uses crt;
type
    TabReal = array[1..5000] of real; { tablica liczb }
           { rzeczywistych }
    PString = ^string; { wskaźnik do łańcucha }

var
    s : PString; { zmienna typu wskaźnik do łańcucha }
    TabTabReal : array[1..100] of ^TabReal; { tablica wskaźników }
    Sterta : pointer; { wskaźnik wysokości sterty }
    i : integer; { pomocniczy licznik }

procedure IlePamieci;
begin
    writeln('Wolne: ', MemAvail, ' max. blok: ', MaxAvail, ' bajtow.');
```

```

end;

begin
    clrscr;
    writeln(s^);           { zmienna nie utworzona }
    new(s);                 { więc ją tworzymy }
    writeln(s^);           { utworzona, lecz nie zainicjalizowana }
    s^ := 'No wreszcie!';   { inicjalizujemy }
    writeln(s^);           { teraz jest OK }
    dispose(s);            { usuwamy }
    writeln(s^);           { zmienna nie została całkowicie zniszczona! }
    mark(Sterta);          { zaznaczamy 'poziom' sterty }
    i := 1;                { tworzymy tablicę tablic dynamicznych }
    while MemAvail > SizeOf(TabReal) do { tyle wierszy ile się da }
    begin
        IlePamieci;        { ile mamy pamięci? }
        new(TabTabReal[i]); { tworzymy nowy wiersz }
        Inc(i);             { zwiększamy indeks wiersza }
    end;
    dispose(TabTabReal[3]); { usuwamy jeden wiersz tablicy }
    IlePamieci;
    release(Sterta);        { zwalniamy hurtem całą pamięć }
    IlePamieci;
    Repeat until Keypressed;
end.

```

Pierwsza część programu demonstruje etapy tworzenia, wykorzystania i usunięcia zmiennej wskazywanej (w naszym przypadku łańcucha) za pomocą procedur new i dispose. Zauważ, że utworzenie zmiennej wskazywanej nie jest równoznaczne z jej inicjalizacją, a po wykonaniu procedury dispose treść łańcucha nie jest niszczone, chociaż może być niekompletna.

Druga część tworzy typową strukturę wielkiej tablicy, przydzielając pamięć dla poszczególnych wierszy, dopóki to jest możliwe. Zauważ, że po usunięciu trzeciego wiersza tablicy na ogół okazuje się, że rozmiar największego dostępnego bloku jest mniejszy od całkowitego rozmiaru wolnego obszaru sterty, co uniemożliwia tworzenie większych struktur dynamicznych. Wreszcie instrukcja release zwalnia całą stertę "wzwyż" począwszy od miejsca zarejestrowanego w zmiennej Sterta.

Zapamiętaj

- 1) Do przechowywania większych ilości danych możesz w Pascalu wykorzystać zmienne wskazywane (dynamiczne).
- 2) Zmienne wskazywane są umieszczane na tzw. stercie (teoretycznie w dowolnym miejscu pamięci). Mogą one być tworzone i niszczone dynamicznie, w zależności od potrzeb.
- 3) Zmienna wskazywana lokalizowana jest za pomocą wskaźnika, który zawiera jej adres (miejsce w pamięci). Wskaźniki mogą wskazywać na zmienne konkretnego typu, mogą też być wskaźnikami amorficznymi (pointer).
- 4) Przed wykorzystaniem zmiennej dynamicznej należy ją utworzyć (procedurą new), a po wykorzystaniu - usunąć (procedurą dispose).
- 5) Do przydzielania i zwalniania bloków pamięci na stercie służą również procedury GetMem, FreeMem, mark i release.

ZADANIE 101.

1) Przepisz instrukcje wraz z wytłumaczeniami związane ze wskaźnikami (new, mark itp.).

2) Wykonaj program i listing wpisz do zeszytu.

Z użyciem zmiennych dynamicznych oblicz obwód oraz pole powierzchni rombu po wczytaniu jego przekątnych. Wykonaj tworzenie tych zmiennych w pamięci oraz po ich wykorzystaniu zwolnij pamięć. Napisz procedurę sprawdzającą ilość całkowitej wolnej pamięci oraz maksymalny wolny blok pamięci. Procedura ta powinna mieć nazwę Pamiec_ile_cztery_pierwsze_litery_nazwiska. Procedura ta powinna dawać wynik np.

cala_pamiec=20038 kB max. blok=12301 KB.

Wywołaj tę procedurę

- przed utworzeniem zmiennych dynamicznych,
- po utworzeniu zmiennych dynamicznych,
- po skasowaniu zmiennych dynamicznych.

Użyj komentarzy pokazujących się na ekranie np.

„Ilość wolnej pamięci przed utworzeniem zmiennych”
 cała_pamiec=20038 kB max. blok=12301 KB.

„Ilość wolnej pamięci po utworzeniu zmiennych”
 cała_pamiec=19938 kB max. blok=11301 KB.

„Ilość wolnej pamięci przed po skasowaniu zmiennych”
 cała_pamiec=20038 kB max. blok=12301 KB.

Zmienne muszą mieć trzy pierwszy litery Twojego nazwiska np. a_kow.

3)Przepisz wygląd ekranu po uruchomieniu programu

Przykład 45a)

Temat Demonstracja Listy jednokierunkowej.

Wykonaj:

- 1)Przepisz temt przykładu.
- 2)Przeczytaj opis działania czterech etapów tworzenia programu.
- 3)Poproś nauczyciela o program(przykład etap4 patrz poniżej) w postaci pliku *.pas. Uruchom ten program. Przerysuj tabelę „Lista dynamiczna do zapamiętywania temperatury notowanej, co godzinę przez stację meteorologiczną” (patrz poniżej).
- 4)Narysuj w zeszycie listę oraz wykonaj opis (głowa, nil). Do czego jest podobna lista. Podaj dwie zalety oraz wadę.
- 5)Co to jest dostęp swobodny do danych.
- 6)Przerysuj wraz z opisem dodawanie elementu do listy
- 7) Przerysuj wraz z opisem usuwanie elementu z listy

Lista jednokierunkowa jako przykład dynamicznej struktury danych.

Lista jednokierunkowa jest bardzo podobna do tablicy. Tak jak tablica służy ona do przechowywania wielu elementów tego samego typu.

Zaleta 1

Jednak w odróżnieniu od tablicy lista może mieć zmienną długość, czyli przechowywać różną ilość elementów w zależności od potrzeb. Własność ta jest bardzo ważna, gdyż często pozwala zaoszczędzić dużo pamięci.

Zaleta2

Drugą zaletą listy jest to, że gdy znajdziemy w niej jakiś element, to można go bardzo łatwo i szybko usunąć.

Wada

Niestety lista ma też swoje wady. Największą z nich jest to, że do elementów listy nie można się odwoływać tak jak do elementów tablicy za pomocą nawiasów kwadratowych, ale aby dotrzeć do danego elementu trzeba najpierw dotrzeć do wszystkich elementów poprzedzających go.

Dostęp swobodny do danych

Taki dostęp do elementów jaki występuje w przypadku tablic (za pomocą nawiasów kwadratowych) nazywa się w informatyce dostępem swobodnym, ponieważ umożliwia szybki (swobodny) dostęp do każdego elementu struktury (w tym przykładzie tablicy). Lista nie oferuje dostępu swobodnego do swoich elementów.

Budowa listy



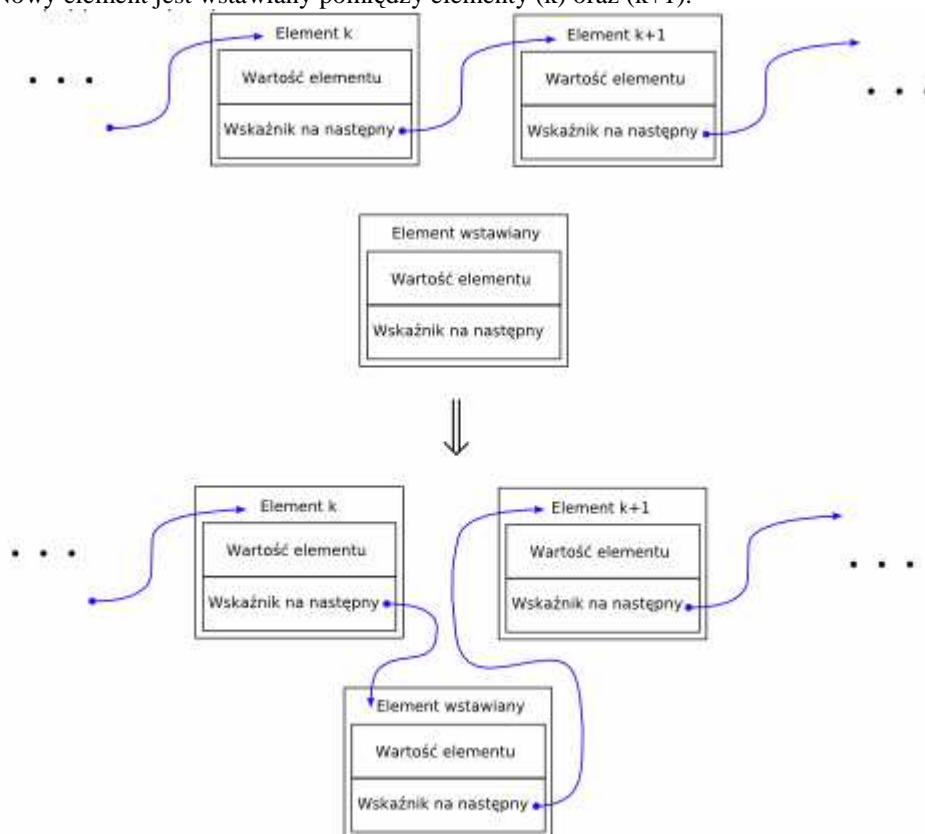
Opis budowy listy:

- 1)Lista składa się z wielu elementów.
- 2)Każdy element listy reprezentowany jest jako pewien typ rekordowy.
- 3)Typ ten zawiera co najmniej dwa pola.
 - Jedno z nich przechowuje wartość, którą chcemy przechowywać na liście,
 - drugie przechowuje wskaźnik na następny element listy.

Tak więc jeżeli wiemy gdzie znajduje się pierwszy element listy, możemy bez problemu dostać się do wszystkich pozostałych (ponieważ pierwszy element pokazuje gdzie znajduje się drugi, drugi element pokazuje gdzie znajduje się trzeci, trzeci pokazuje gdzie znajduje się czwarty i tak dalej, aż do ostatniego elementu listy). Dlatego właśnie musimy w pewnej zmiennej w programie zapisać gdzie w pamięci znajduje się pierwszy element listy (tak zwana **głowa listy**). Na powyższym rysunku zmienna ta oznaczona jest jako "wskaźnik na głowę". Nie będziemy potrzebować w programie więcej zmiennych do przechowywania kolejnych elementów listy. Kolejne elementy listy będą tworzone za pomocą funkcji New, a ich adresy będą zapisywane w polach wskaźnikowych, w którymś z już utworzonych elementów. Ostatni element listy nie ma na co pokazywać (nie ma już następnego elementu), tak więc ustawiamy mu wartość **wskaźnika na nil**. W ten sposób można łatwo sprawdzić, czy dany element jest ostatnim, czy nie.

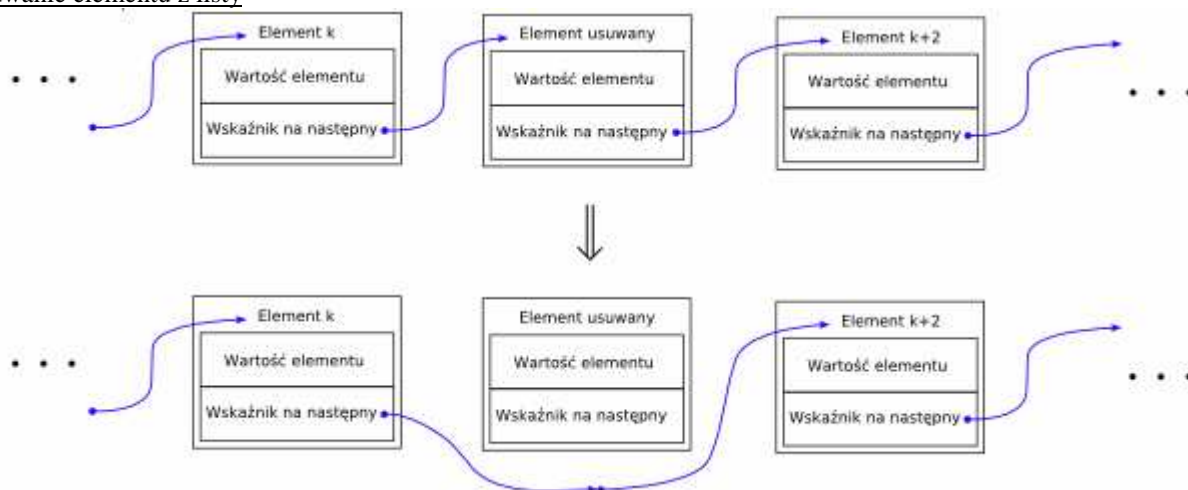
Dodawanie elementu do listy

Na poniższym rysunku znajduje się przedstawiony schematycznie proces dodawania nowego elementu do listy. Nowy element jest wstawiany pomiędzy elementy (k) oraz (k+1).



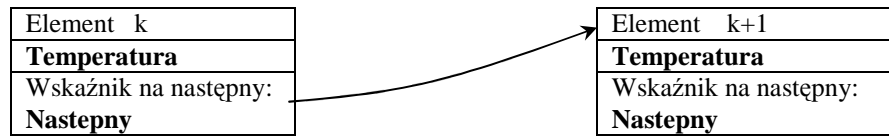
Dodawanie nowego elementu do listy jest bardzo intuicyjne. Wystarczy ustawić wskaźnik na następny element dla dodawanego elementu na element, przed którym go wstawiamy (czyli (k+1)), a następnie wskaźnik w poprzednim elemencie na dodawany. Uważaj, żebyś nie zamienił kolejności ustawiania tych dwóch wskaźników, gdyż wtedy kod nie będzie działał

Usuwanie elementu z listy



Usuwanie elementu z listy zostało przedstawione na poniższym rysunku. Również ta operacja jest intuicyjna - polega na ustawieniu wskaźnika znajdującego się w poprzednim elemencie tak, aby pokazywał na jeden element dalej, czyli na element znajdujący się bezpośrednio za usuwanym.

Lista dynamiczna do zapamiętywania temperatury notowanej, co godzinę przez stację meteorologiczną.



Etap1

Temat: Zadeklarowanie elementu listy. Inicjacja pustej listy → czyli ustawienie wskaźnika na nil, ponieważ na początku działania programu lista jest pusta, zmienna ta musi być ustawiona na nil.

Opis działania programu:

Typ Welement jest typem wskaźnikowym, który służy do przechowywania adresów zmiennych typu Element_listy. Typ Element_listy jest natomiast typem reprezentującym pojedynczy element listy.

Uważaj na kolejność! Jeżeli zadeklarujesz typy Element_listy i Welement w odwrotnej kolejności, Twój program może się nie skompilować.

```
Program demonstracja_listy;
uses crt;

type
  Welement      = ^Element_listy;

  Element_listy = record
    temperatura  : integer;
    nastepny     : Welement;
  end;

var
  lista: Welement;

begin
  clrscr;
  lista:=nil;

repeat until keypressed;
end.
```

Etap2

Temat: Dodanie elementu do listy.

Do programu została dopisana procedura **dodaj**.

Procedura ta działa następująco. Najpierw deklarowana jest zmienna wskaźnikowa tmp, wskazująca w pamięci adres, pod którym mogą się znajdować jakieś dane typu Element_listy. Następnie przypisujemy nowo utworzonemu elementowi wartość.

Pamiętaj, że zmienna tmp sama w sobie nie przechowuje danych typu Element_listy. Ona tylko pokazuje, gdzie te dane się znajdują. Aby faktycznie można było w tym miejscu zachować jakieś dane, trzeba to miejsce zarezerwować, co dokonuje się poprzez wywołanie funkcji New.

Następnie ustalamy wartości odpowiednich wskaźników, w taki sposób, aby nowo tworzony element znalazł się na liście za elementem podanym jako parametr poprzednik. Należy przy tym uważać przy obsługiwaniu przypadku, kiedy lista=nil (na samym początku) - wtedy początkiem listy staje się po prostu tworzony element listy. Ponieważ najłatwiej dodać nowy element na początku listy, wywołanie procedury dodaj będzie miało w większości przypadków postać:

```
dodaj(lista, jakas_wartosc);
```


gdzie lista oznacza zadeklarowany wcześniej wskaźnik na głowę listy.

```

Program demonstracja_listy;
uses crt;

type
  Welement      = ^Element_listy;

  Element_listy = record
    temperatura : integer;
    nastepny    : Welement;
  end;

var
  lista: Welement;

procedure dodaj(poprzednik: Welement; nowa_wartosc: Integer);
var
  tmp: Welement;
begin
  New(tmp);
  tmp^.temperatura := nowa_wartosc;
  if poprzednik = nil then
    begin
      lista := tmp;
      tmp^.nastepny := nil;
    end
  else
    begin
      tmp^.nastepny := poprzednik^.nastepny;
      poprzednik^.nastepny := tmp;
    end;
  end;

begin
  clrscr;
  lista:=nil;
  dodaj(lista,15);

repeat until keypressed;
end.

```

Etap3

Temat: Usuwanie elementów z listy.

Powyższa procedura usuwa element znajdujący się za elementem przekazanym w parametrze jako poprzednik. Na początku zapamiętuje ona adres elementu znajdującego się za elementem usuwanym. Adres ten musi być zapamiętany, gdyż w następnej linijce miejsce zajmowane przez usuwany element jest zwalniane przy użyciu funkcji Dispose, co wiąże się z utratą tej danej. Na samym końcu zapamiętany adres podstawiany jest do wskaźnika znajdującego się w elemencie poprzedzającym usuwany.

```

Program demonstracja_listy;
uses crt;

type
  Welement      = ^Element_listy;

  Element_listy = record
    temperatura : integer;
    nastepny    : Welement;
  end;

var

```

```

    lista: Welement;

procedure dodaj(poprzednik: Welement; nowa_wartosc: Integer);
var
    tmp: Welement;
begin
    New(tmp);
    tmp^.temperatura := nowa_wartosc;
    if poprzednik = nil then
    begin
        lista := tmp;
        tmp^.nastepny := nil;
    end
    else
    begin
        tmp^.nastepny := poprzednik^.nastepny;
        poprzednik^.nastepny := tmp;
    end;
end;

procedure usun(poprzednik: Welement);
var
    tmp: Welement;
begin
    if poprzednik^.nastepny <> nil then
    begin
        tmp := poprzednik^.nastepny^.nastepny;
        Dispose(poprzednik^.nastepny);
        poprzednik^.nastepny := tmp;
    end;
end;

begin
    clrscr;
    lista:=nil;
    dodaj(lista,15);
    usun(lista);
    repeat until keypressed;
end.

```

Etap4

Temat: Wyszukiwanie elementu z listy.

Funkcja najpierw ustawia wartość zmiennej tmp na początek listy, a następnie wykonuje kod w pętli tak długo, aż zmienna tmp nie osiągnie wartości nil, co oznacza że doszliśmy do końca listy. Jeżeli w trakcie wykonywania pętli znaleziona zostanie szukana wartość, to wynik funkcji ustawiany jest na true.

```

Program demonstracja_listy;
uses crt;

type
    Welement      = ^Element_listy;

    Element_listy = record
        temperatura : integer;
        nastepny     : Welement;
    end;

var
    lista: Welement;
    szukana_temperatura: integer;

```

```

procedure dodaj(poprzednik: Welement; nowa_wartosc: Integer);
var
  tmp: Welement;
begin
  New(tmp);
  tmp^.temperatura := nowa_wartosc;
  if poprzednik = nil then
    begin
      lista := tmp;
      tmp^.nastepny := nil;
    end
    else
    begin
      tmp^.nastepny := poprzednik^.nastepny;
      poprzednik^.nastepny := tmp;
    end;
end;

procedure usun(poprzednik: Welement);
var
  tmp: Welement;
begin
  if poprzednik^.nastepny <> nil then
    begin
      tmp := poprzednik^.nastepny^.nastepny;
      Dispose(poprzednik^.nastepny);
      poprzednik^.nastepny := tmp;
    end;
end;

function istnieje(lista: Welement; wartosc_szukana: Integer):boolean;
var
  tmp: Welement;
begin
  tmp := lista;
  istnieje := false;
  while tmp <> nil do
    begin
      if tmp^.temperatura = wartosc_szukana then
        istnieje := true;
        tmp := tmp^.nastepny;
      end;
    end;
end;

begin
  clrscr;
  lista:=nil;
  dodaj(lista,5);

  dodaj(lista,16);
  write('Podaj temperatur@=');readln(szukana_temperatura);
  if istnieje(lista,szukana_temperatura)=true then
    begin
      Writeln('Temperatura ',szukana_temperatura,' istnieje');
    end
    else
    begin
      Writeln('Temperatura ',szukana_temperatura,' nie istnieje');
    end;

  usun(lista);

```

```
repeat until keypressed;
end.
```

ZADANIE 102.

Na podstawie treści programu otrzymanego od nauczyciela. Napisz program:

miesiąc urodzenia	pola (bez polskich liter)
Styczeń	waga, nazwisko
Luty	numer, ulica
Marzec	wiek, imię
Kwiecień	cena, towar
Maj	masa_atomowa, pierwiastek
Czerwiec	wysokość, budynek
Lipiec	prędkość, pojazd
Sierpień	szerokość, ulica
Wrzesień	masa, towar
Październik	droga, między
Listopad	siła, działanie
Grudzień	rocznik, samochód

Założenia do bazy danych:

- pierwsze pole jest numeryczne, drugie pole jest tekstowe
- nazwa listy → list_cztery_pierwsze_litery_nazwisk,
- nazwa pola rekordu → znacząca_nazwa_dwie_pierwsze_litery_nazwisk np. waga_ko
- Opcje wykonywane przez program
 - Wypisanie danych na listę
 - Dopisywanie danych na listę
 - Kasowanie danych z listy
 - sprawdzanie istnienie danych na liście
 - sprawdzanie istnienie danych na liście

Przykład 45b)

Zapisz pytania i odpowiedzi po nimi

1) Co to jest kolejka oraz skrót nazwy (angielskie wy tłumaczenie).

2) Co to jest stos oraz skrót nazwy. (angielskie wy tłumaczenie)

3) Narysuj drzewo

-gdzie jest wykorzystywane

-co przyspieszają

-jakie dziedziny

4) Budowa drzewa

-długością

-poziom węzła.

-dzieci

-liść.

-wierzchołek

-drzewa binarne,

5) Wymień oraz opisz operacje na drzewach

6) Opisz pojęcia:

-preorder

-postorder

-inorde

Kolejka i stos

Istnieją dwa szczególne rodzaje listy: kolejka i stos.

Kolejka charakteryzuje się tym, że elementy dodawane są zawsze na początku, a usuwane tylko i wyłącznie z jej końca. W literaturze kolejki określane są jako FIFO (First In First Out).

W **stosie** obie operacje dodania oraz usunięcia elementu zawsze odnoszą się do początku i jego pierwszego elementu. W literaturze stos określa się jako LIFO (Last In First Out).

Drzewo źródło: Wikipedia, (wolnej encyklopedi)

Drzewo uwagi wstępne:

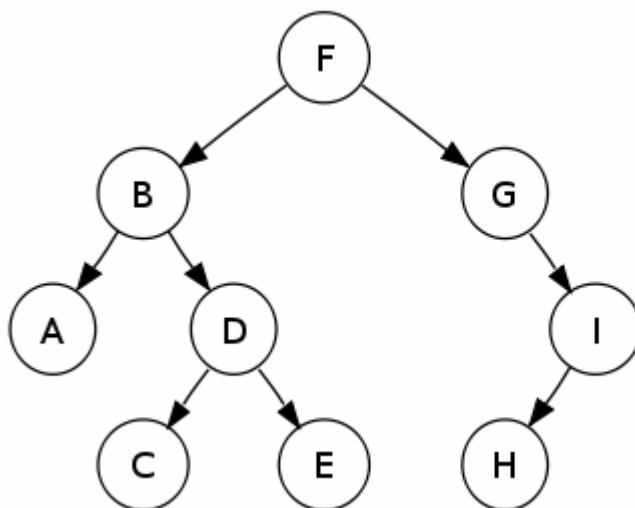
W naturalny sposób reprezentuje hierarchię danych (obiektów fizycznych i abstrakcyjnych, pojęć, itp.) jest więc stosowane głównie do tego celu.

Drzewa ułatwiają i przyspieszają wyszukiwanie, a także pozwalają w łatwy sposób operować na posortowanych danych.

Drzewa są stosowane praktycznie w każdej dziedzinie informatyki (np. bazy danych, grafika komputerowa, przetwarzanie tekstu, telekomunikacja).

Budowa drzewa

Drzewa składają się z **wierzchołków (węzłów)** oraz łączących je **krawędzi**. Jeśli drzewo nie jest puste, tzn. liczba wierzchołków jest większa od zera, jeden z nich jest wyróżniony i nazywany **korzeniem drzewa**; na rysunku F.



Ciąg krawędzi łączących węzły nazywa się **ścieżką**. Istnieje dokładnie jedna ścieżka łącząca korzeń z wszystkimi pozostałymi wierzchołkami.

Liczba krawędzi w ścieżce od korzenia do węzła jest nazywana **długością** – liczba o jeden większa określa **poziom węzła**. Wysokością drzewa jest największy poziom istniejący w drzewie. Np. korzeń znajduje się na 1. poziomie, węzły A, D i I na poziomie 3.; wysokość drzewa to 4.

Wszystkie wierzchołki połączone z danym wierzchołkiem, a leżące na następnym poziomie są nazywane **dziećmi** tego węzła (np. dziećmi wierzchołka F są B i G, natomiast wierzchołka B: A i D). Wierzchołek może mieć dowolną liczbę dzieci, jeśli nie ma ich wcale nazywany jest **liściem**. Liśćmi w przykładowym drzewie są A, C, E, H.

Wierzchołek jest rodzicem dla każdego swojego dziecka. Każdy węzeł ma dokładnie jednego rodzica, wyjątkiem jest korzeń drzewa, który nie ma rodzica.

Specjalne znaczenie w informatyce mają **drzewa binarne**, w których liczba dzieci ograniczona jest do dwóch. Szczególnie popularne są BST i ich różne odmiany, np. drzewa AVL, drzewa czerwono-czarne.

Drzewa które posiadają więcej niż dwoje dzieci są nazywane drzewami wyższych rzędów, np. drzewo trie, B-drzewo.

Operacja na drzewach

Podstawowe operacje na drzewach to:

- wyliczenie wszystkich elementów drzewa,
- wyszukanie konkretnego elementu,
- dodanie nowego elementu w określonym miejscu drzewa,
- usunięcie elementu.

Pod pojęciem przechodzenia drzewa rozumie się odwiedzanie kolejnych wierzchołków, zgodnie z zależnościami rodzic-dziecko. Jeśli przy przechodzeniu drzewa na wierzchołkach są wykonywane pewne działania, to mówi się wówczas o przechodzeniu:

- preorder - gdy działanie jest wykonywane najpierw na rodzicu, następnie na dzieciach;
- postorder - gdy działanie jest wykonywane najpierw na wszystkich dzieciach, na końcu na rodzicu.

W przypadku drzew binarnych istnieje jeszcze metoda **inorder**, gdzie najpierw wykonywane jest działanie na jednym z dzieci, następnie na rodzicu i na końcu na drugim dziecku.

Jeśli działaniem byłoby wypisanie liter przechowywanych w węzłach przykładowego drzewa, przy przechodzeniu drzewa metodą preorder otrzymamy F, B, A, D, C, E, G, I, H, przy przechodzeniu drzewa metodą postorder: A, C, E, D, B, H, I, G, F.

Procedury funkcje i typy modułu DOS

PROCEDURY

a)dotyczące dysków

DISKFREE(D)–podaje liczbę wolnych bajtów na dysku D–przyjmuje:

0 – dysk bieżący

1 – dysk A

2 – dysk B itp.

DISKSIZE(D)–podaje rozmiar dysku

b)dotyczące katalogów

GETDIR(D,S)–odczytuje nazwę bieżącego katalogu na dysku do zmiennej S o typie STRING

CHDIR(ŚCIEŻKA)–zmiana katalogu z bieżącego na określonego w ŚCIEŻCE

MKDIR(KATALOG)–utworzenie katalogu o nazwie znajdującej się w zmiennej KATALOG

RMDIR(KATALOG)–kasowanie katalogu o nazwie znajdującej się w zmiennej KATALOG

c)dotyczące atrybutów

GETFATTR(ZBIOR, ATRYBUTY)–odczytanie atrybutów ZBIORU i zapisanie wyniku w zmiennej ATRYBUTY. ZBIÓR musi być skojarzony z plikiem, którego odczytujemy atrybuty lecz nie może być otwarty.

* ReadOnly (1) - zbiór tylko do odczytu

* Hidden (2) - zbiór ukryty

* SysFile (4) - zbiór systemowy

* VolumeID (8) - etykieta dysku

* Directory (16) - katalog

* Archive (32) - zbiór archiwalny

* AnyFile (63) - dowolny zbiór

SETFATTR(ZBIOR, ATRYBUTY)–nadanie atrybutów ZBIOROWI zapisanych w zmiennej ATRYBUTY. ZBIÓR musi być skojarzony z plikiem, któremu nadajemy atrybuty lecz nie może być otwarty.

d)dotyczące daty i czasu

GETTIME(H,M,S,S1)–odczytuje czas systemowy zapisując do zmiennych H–godzina,M–minuta,S–sekunda,S1–setne sekundy

SETTIME(H,M,S,S1)–nadanie czas systemowego zapisując H–godzina,M–minuta,S–sekunda,S1–setne sekundy

GETDATE(ROK, MIESIĄC, DZIEŃ, DZIEŃ_TYG)–odczytanie daty systemowej i zapisanie w zmiennych ROK, MIESIĄC, DZIEŃ, DZIEŃ_TYG

SETDATE(ROK, MIESIĄC, DZIEŃ)–ustawienie daty systemowej określonej w zmiennych ROK, MIESIĄC, DZIEŃ

e)dotyczące zbiorów

FINDFIRST(S,A,R)–odnajduje na dysku plik o nazwie S (dopuszczalne * oraz ?) i atrybucie A ,dane tego pliku zapisuje w rekordzie R typu SEARCHREC o polach:

pole1 FILL : array[1..21] of byte

pole2 ATTR : byte

pole3 TIME : longint

pole4 SIZE : longint

pole5 NAME : string[12]

w polu1 zapisywane są dane wykorzystywane przez system

w polu2 zapisywane są dane o atrybutach

–readonly \$01 uwaga:\$ oznacza ,że liczby są w

–hidden \$02 podawane heksagonalnie

–sysfile \$04

–volumeid \$08

–directory \$10

–archive \$20

–anyfile \$3F

w polu3 zapisywane są dane o czasie utworzenia(modyfikacji)

w polu4 zapisywany jest rozmiar pliku

w polu5 zapisywana jest jego nazwa

FINDNEXT(R)–odnajduje na dysku kolejny plik opisany rekordem R

SETFTIME((ZBIOR, DATA_CZAS)–nadanie daty i czasu ZBIOROWI zapisanych w zmiennej DATA_CZAS. ZBIÓR musi być skojarzony z plikiem, któremu nadajemy lecz nie może być otwarty. Przekształcenie daty i czasu do postaci upakowanej odbywa się za pomocą instrukcji PACKTIME a czynność odwrotna UNPACKTIME

ZMIENNE

DIRSTR – zmienna przechowująca nazwę katalogu

SEARCHREC – rekord do opisu pliku

DOSERROR – zmienna podająca jaki był błąd przy wywołaniu funkcji modułu DOS

- 0 – bez błędu
- 2 – nie znaleziono katalogu
- 3 – błędna ścieżka
- 5 – zabroniony dostęp
- 6 – błędny uchwyt
- 18 – nie znaleziono zbioru

PRZYKŁAD 46

Wykonaj:

- 1)Przepisz temat w zeszycie
- 2)Uruchom program (Pamiętaj o zmianie ścieżki dostępu do grafiki)
- 3)Wpisz listing do zeszytu.
- 4)Zapisz w zeszycie nowe instrukcje użyte w przykładzie wraz z wytłumaczeniem.

Temat:Program do wyświetlania sekundnika

```
PROGRAM STRING4;
USES
  CRT,DOS,GRAPH;
VAR
  GODZINY,MINUTY,SEKUNDY,SEKUNDY100:WORD;
  S:STRING;
  STEROWNIK,TRYB:INTEGER;
BEGIN
  STEROWNIK:=DETECT;
  INITGRAPH(STEROWNIK,TRYB,'C:\TP\BGI');
  SETCOLOR(0);
  SETTEXTSTYLE(3,0,4);
  REPEAT
    GETTIME(GODZINY,MINUTY,SEKUNDY,SEKUNDY100);
    STR(SEKUNDY,S);
    BAR(30,30,200,200);
    OUTTEXTXY(100,100,S);
  UNTIL KEYPRESSED;
END.
```

ZADANIE 103.

Dokonaj modyfikacji programu z przykładu poprzedniego tak aby wyświetlany był cały czas. Godziny, minuty, sekundy, setne sekund. Wszystkie pola czasu powinny być oddzielone dwukropkami.

PRZYKŁAD 47

Wykonaj:

1)Przepisz temat w zeszycie

2)Na dowolnym dysku zalożyć folder Twoje nazwisko. Do tego folderu wgrać 15 dowolnych plik tekstowy. Zmień atrybuty temu plikowi na R, S, H, A we wszystkich możliwych kombinacjach np. ASH czy AR (kombinacji jest 15)

sprawdzanie atrybutów w DOS

ATTRIB *.*

ustawianie atrybutów w DOS

ATTRIB akcja.txt -R -H +S

Ustawienie atrybut S i odebranie R H

3)Uruchom program

-sprawdź atrybuty folderu założonego w punkcie 2→zapisz w zeszycie jaka liczba została wyświetlona oraz co to oznacza

-sprawdź atrybuty wszystkich plików (piętnastu)→zapisz w zeszycie jaka liczba została wyświetlona oraz co to oznacza

np.

1 atrybut R

....

39 atrybuty ASHR

.....

4)Wpisz listing do zeszytu.

5)Zapisz w zeszycie nowe instrukcje Pascal użyte w przykładzie wraz z wytłumaczeniem (łącznie z atrybutami) oraz instrukcje DOS dotyczące atrybutów.

Uwaga:

Przy wpisywaniu folderów czy plików, których atrybuty chcesz sprawdzić musisz wpisać łącznie ze ścieżką dostępu np.

C:\KOWALSKI dla folderu

C:\KOWALSKI\ANIMACJA.BAK dla pliku

Temat: Program do określania atrybutów plików oraz folderów.

PROGRAM OKRESLANIE_ATRYBUTOW;

USES

CRT,DOS;

VAR

ATRYBUTY : WORD;

ZBIOR_DYSK : STRING;

ZBIOR_SK : FILE;

DALEJ : CHAR;

BEGIN

REPEAT

CLRSCR;

WRITELN('PODAJ ZBIÓR ŁĄCZNIE ZE ŚCIEŻKĄ DOSTĘPU ');

WRITE('KTÓREGO CHCESZ SPRAWDZIĆ ATRYBUTY : ');

READLN(ZBIOR_DYSK);

ASSIGN(ZBIOR_SK, ZBIOR_DYSK);

GETFATTR(ZBIOR_SK,ATRYBUTY);

WRITELN('ATRYBUTY=',ATRYBUTY);

WRITELN('CZY CHCESZ PONOWNEGO SPRAWDZENIA T/N');

DALEJ:=READKEY;

UNTIL (DALEJ='n')OR(DALEJ='N');

END.

ZADANIE 104.

Wykonaj:

1)Przepisz temat w zeszycie

Temat Program do wyświetlania i zmiany atrybutów.

Napisz program do nadawania atrybutów (wzoruj się na przykładzie poprzednim). Wykonaj według algorytmu.

KROK1: Program spyta się o plik do zmiany atrybutów wraz ze ścieżką dostępu.
 KROK2: Program wyświetli aktualne atrybuty pliku.
 KROK3: Program będzie pytał się, jakie atrybuty ma mieć plik.
 KROK4: Program zmieni atrybuty wg naszego życzenia.
 KROK5: Po zamianie, atrybuty zbioru będą również wyświetlane.

Pisanie programów wywoływanych z parametrami.

W PASCALU programy mogą być pisane tak, że ich wywołanie będzie możliwe z parametrami. Występuje tutaj bardzo duże podobieństwo (przy wywoływaniu) z plikami wsadowymi uruchamianymi z parametrami.

np. SZUKAJ C:\TP\ZBIOR.PAS WRITELN

Program SZUKAJ wywołany został z parametrami (dwoma)

pierwszy → C:\TP\ZBIOR.PAS

drugi → WRITELN

Do pisania programów z parametrami potrzebne są następujące instrukcje:

PARAMETR(k):STRING – napis równy k – temu parametrowi wywołania programu z poziomu DOS lub k – tej części wartości parametru o nazwie PARAMETERS Turbo Pascalu

PARAMCOUNT:WORD – liczba parametrów z jakimi wywołujemy program

UWAGA:

Gdy chcesz testować program z parametrami to powinieneś wpisać parametry przed jego uruchamianiem. W tym celu w opcji RUN wybierz podopcję Parameters... po jej uaktywnieniu wpisz parametry oddzielone spacjami na koniec OK. Teraz możesz uruchomić program.

PRZYKŁAD 48

Wykonaj:

- 1) Zapisz w zeszycie teorię „Pisanie programów wywoływanych z parametrami” wraz z uwagą.
- 2) Przepisz temat w zeszycie oraz treść zadania.
- 3) Wpisz treść zadania w Pascalu.
- 4) Uruchom program z różnymi kombinacjami parametrów. Jak wpisywać parametry w Pascalu patrz UWAGA w teorii.
- 5) Wpisz listing do zeszytu.

Temat: Program prezentujący sposób pisania programów wywoływanych z parametrami.

Treść zadania

Program może być wywoływany z parametrami

– gdy nie ma parametru to poinformuje o tym

– gdy parametrem jest ? to otrzymamy wytłumaczenie

– gdy liczba to zostanie ona wyświetlona na ekranie

– gdy parametrem jest znak nie będący liczbą to zostaniemy poinformowani, że konwersja nie udała się

```
PROGRAM WYWOŁANIE_Z_PARAMETRAMI;
```

```
USES
```

```
  CRT,DOS;
```

```
VAR
```

```
  I,K:INTEGER;
```

```
  A:REAL;
```

```
PROCEDURE PARAMETRY(PARAMETR:STRING);
```

```
  BEGIN
```

```
    IF PARAMETR='?' THEN
```

```
      WRITELN('DEMONSTRACJA WYKONANIA PROGRAMU Z PARAMETRAMI ');
```

```
  END;
```

```
  BEGIN
```

```
    CLRSCR;
```

```
    IF PARAMCOUNT=0 THEN {GDY PROGRAM JEST WYWOŁANY BEZ PARAMETRU}
```

```
      WRITELN('PROGRAM MUSI MIEĆ PARAMETRY ')
```

```
    ELSE
```

```
  BEGIN
```

```
    IF PARAMSTR(1)<>'?' THEN {GDY PARAMETREM JEST ?}
```

```
      BEGIN
```

```

VAL(PARAMSTR(1),A,K); {ZAMIANA TEKSTU NA LICZBĘ}
IF K=0 THEN {SPRAWDZENIE POPRAWNOŚCI ZAMIANY}
BEGIN
    WRITELN('KONWERSJA POPRAWNA ');
    WRITELN('WYWOŁAŁEŚ PROGRAM ZE ZMIENNA A=',A:6:2);
END
ELSE
    WRITELN('KONWERSJA NIEPOPRAWNA ');
END;
END;
FOR I:=1 TO PARAMCOUNT DO {PRZYKŁAD ZASTOSOWANIA PROCEDURY}
    PARAMTRY(PARAMSTR(I)); {KTÓREJ PARAMETREM SĄ PARAMETRY }
REPEAT UNTIL KEYPRESSED; {PROGRAMU}
END.

```

ZADANIE 105.

Wykonaj:

- 1)Przepisz temat w zeszycie
- 2)Dokonaj kompilacji programu na EXE.
- 3)Dokonaj testowania programu z różnymi parametrami w DOS czyli z104.exe koko 3
- 4)Jeśli ze wszystkimi parametrami OK. to wpisz listing do zeszytu.

Temat: Napisz program, który będzie:

- Program będzie wywoływany z dwoma parametrami.
- Można go uruchomić po podaniu prawidłowego hasła jako pierwszego parametru.
- Program będzie zamieniał kilobajty na bity. Drugim parametrem będzie ilość kilobajtów.
- Gdy pierwszy parametrem ? program poda pomoc. → "Program do zamiany kilobajtów na bity"
- Program powinien również uwzględniać sytuację gdy będą mniej niż dwa parametry przy jego wywołaniu. → „Program powinien być wywołany z parametrami”

PRZYKŁAD 49

Wykonaj:

- 1)Przepisz temat w zeszycie
- 2)W treści zadania (listingu) zmień na ścieżkę dostępu tam gdzie masz Pascal czyli dysk C folder UCZEN i podfolder TP. Po zmianie uruchom program. Przepisz do zeszytu cztery linijki wyglądu ekranu po uruchomieniu programu.
- 3)Wpisz listing do zeszytu.
- 4)Zapisz w zeszycie nowe instrukcje użyte w przykładzie wraz z wytłumaczeniem. (Findfirst, Findnext, Doserror)

Temat: Program wypisujący wszystkie zbiory (czyli *.*) z określonego dysku oraz określonego katalogu z atrybutem A.

```

PROGRAM WYSWIETLAJ_ZBIORY;
USES
    CRT,DOS;
VAR
    I:INTEGER;
    ATRYBUT:CHAR;
    INFO_O_ZBIORZE:SEARCHREC;{SPRAWDŹ W INSTRUKCJI JAKIE }
BEGIN
    {POLA MA ZMIENNA TYPU REKORD SEARCHREC}
    CLRSCR;
    FINDFIRST('C:\*.*',ANYFILE,INFO_O_ZBIORZE);
    WHILE DOSERROR<>18 DO {DOSERROR RÓWNA SIĘ 18 GDY NIE MA PLIKU}
    BEGIN
        {FINDNEXT NIE ZNAJDZIE ZBIORU}
        IF (INFO_O_ZBIORZE.ATTR AND ARCHIVE) <>0 THEN
            ATRYBUT:='A'
        ELSE
            ATRYBUT:=' ';
        WRITELN(INFO_O_ZBIORZE.NAME:13,INFO_O_ZBIORZE.SIZE:8,' ',ATRYBUT);
        FINDNEXT(INFO_O_ZBIORZE);
    END;

```

```
REPEAT UNTIL KEYPRESSED;
END.
```

ZADANIE 106.

Zmodyfikuj poprzedni program tak aby działał w trybie graficznym.

Wskazówka

-Uruchom tryb graficzny
 -Zamiast instrukcji WRITELN(INFO_O_ZBIORZE.NAME:13,INFO_O_ZBIORZE.SIZE:8,' ',ATRYBUT);
 Zastosuj OUTTEXTXY z odpowiednimi parametrami. Pamiętaj jednak, że musisz zamienić wielkość pliku ze zmiennej liczbowej na zmienną tekstową instrukcją STR.

ZADANIE 107.

Zmodyfikuj przykład tak aby wyświetlane były wszystkie atrybuty.

PRZYKŁAD 50

Wykonaj:

- 1)Przepisz temat w zeszycie
- 2)Uruchom program
- 3)Wpisz listing do zeszytu.
- 4)Zapisz w zeszycie zakres zmiennej byte, longint, word. Na podstawie Internetu.
- 5)Zapisz w zeszycie nowe instrukcje użyte w przykładzie wraz z wytłumaczeniem. (CHDIR, GETDIR, MKDIR, DISKFREE, DISKSIZE)

Temat: Program do obsługi katalogów i całych dysków

```
PROGRAM DANE_SYS;
USES
  CRT,DOS;
VAR
  SCIEZKA, KATALOG, OPCJA:STRING;
  WOLNE, POJEMNOSC:LONGINT;
  K, LICZBA:INTEGER;
PROCEDURE ZMIANA_KATALOGU;
BEGIN
  WRITE('PODAJ ŚCIEŻKĘ DOSTĘPU ORAZ NAZWĘ KATALOGU : ');
  READLN(SCIEZKA);
  CHDIR(SCIEZKA);
  GETDIR(0,KATALOG);
  WRITELN('AKTUALNY KATALOG : ',KATALOG);
END;
PROCEDURE ZAKLADANIE_KATALOGU;
BEGIN
  WRITE('PODAJ ŚCIEŻKĘ DOSTĘPU ORAZ NAZWĘ KATALOGU DO
ZAŁOŻENIA : ');
  READLN(SCIEZKA);
  MKDIR(SCIEZKA);
END;
PROCEDURE KASOWANIE_KATALOGU;
BEGIN
  WRITE('PODAJ ŚCIEŻKĘ DOSTĘPU ORAZ NAZWĘ KATALOGU DO
KASOWANIA : ');
  READLN(SCIEZKA);
  RMDIR(SCIEZKA)
END;
BEGIN
  CLRSCR;
  WOLNE:=DISKFREE(0);
  WRITELN('WOLNEGO NA DYSKU JEST : ',WOLNE,' BAJTÓW');
  POJEMNOSC:=DISKSIZE(0);
  WRITELN('POJEMNOŚĆ DYSKU JEST : ',POJEMNOSC,' BAJTÓW');
  GETDIR(0,KATALOG);
  WRITELN('AKTUALNY KATALOG : ',KATALOG);
```

```

WINDOW(1,12,79,24);
REPEAT
    CLRSCR;
    WRITELN('MASZ DO WYBORU OPCJE');
    WRITELN('1-ZMIANA KATALOGU');
    WRITELN('2-ZAKŁADANIE KATALOGU');
    WRITELN('3-KASOWANIE KATALOGU');
    WRITE('WYBIERZ OPCJE : ');
    OPCJA:=READKEY;
    VAL(OPCJA,LICZBA,K);
UNTIL K=0;
WINDOW(1,1,79,25);
CLRSCR;
CASE LICZBA OF
    1:ZMIANA_KATALOGU;
    2:ZAKŁADANIE_KATALOGU;
    3:KASOWANIE_KATALOGU;
END;
REPEAT UNTIL KEYPRESSED;
END.

```

ZADANIE 108.

Wykonaj:

- 1)Przepisz temat w zeszycie
- 2)Na dysku C w folderze wasze nazwisk(bez polskich liter do ośmiu znaków) podfolder o nazwie KASUJ. W folderze tym zakładasz cztery pliki tekstwe o nazwach plik1.txt, plik2.txt, plik3.txt, plik4.txt w każdym pliku wpisz nazwisko ucznia.
plik1.txt → atrybut A
plik2.txt, → atrybut S
plik3.txt, → atrybut H
plik4.txt → atrybut R
Wykonaj kopię folderu KASUJ jako KASUJ1 jako kopiowanie całego folderu.
- 3)Napisz program
- 4)Wpisz listing do zeszytu.

Temat: Program z parametrem do usuwania katalogu.

Wskazówka

- Parametrem będzie nazwa katalogu wraz ze ścieżką dostępu.
- Usuwanie katalogi nie mogą mieć podkatalogów.
- Atrybuty zbiorów w usuwanym katalogu nie będą przeszkodą w usunięciu katalogu.

Program rozwiązuj etapami:

- tak aby program wyświetlał parametr,**
- wyświetlanie aktualnego katalogu,**
- przejdźcie do folderu c:\nazwisko_ucznia**
- wyświetlanie aktualnego katalogu,**
- odbierz wszystkim plikom wszystkie atrybuty użycie Finferst oraz Findnext.**

ZADANIA NA SPRAWDZIAN

ZADANIE 1

Napisz program ,który po wczytaniu tekstu do zmiennej drukował będzie ten tekst na ekranie litera po literze w taki sposób ,że każda litera tekstu pisana jest w kolumnie o jeden większej i w wierszu o jeden większym. Gdy skończą się kolumny ekranu to następuje zmniejszanie kolumn itd.

ZADANIE 2

Napisz program ,który będzie wstawiał tekst1 do tekstu2 za wszystkimi wystąpieniami określonej litery w tekście2 . Do program wczytujemy jako dane: tekst1 jako zmienną string , tekst2 jako zmienna string oraz literę za ,którą mamy wstawiać tekst1 w tekście2.

ZADANIE 3

Napisz program symulującego pracę stopera. Stoper będzie uruchamiany na wciśnięcie dowolnego klawisza jego wyłączenie będzie na wykonywane na tej samej zasadzie. Kasowanie stopera odbywać się będzie na wciśnięcie klawisza ESC. Program będzie pracował w trybie graficznym.

ZADANIE 4

Napisz program rozwiązujący równanie kwadratowe. Program będzie wywoływany z trzema parametrami– współczynniki postaci ogólnej równania kwadratowego. Program z parametrem ? poda pomoc. Program powinien również uwzględniać sytuację gdy będą mniej niż trzy parametry przy jego wywołaniu.

ZADANIE 5

Napisz program działający w trybie graficznym symulujący instrukcję DIR/p

BŁĘDY

numer błędu	Znaczenie
2	a)w nazwie programu nie może być znaku – (minus) b)po VAR lub USES napisałeś ; (średnik) c)nie napisałeś CRT po USES
3	a)używasz nie zadeklarowanej zmiennej w VAR b)źle napisałeś instrukcję (pomyłka w pisowni) c)w instrukcji WRITELN(zapomniłeś napisać po (' d)nie napisałeś deklaracji USES CRT; w programie
4	w programie występują dwie takie same nazwy np. procedur lub program nazywa się jak procedura itp.
5	zamiast ' napisałeś `
8	nieparzysta liczba apostrofów w linii
10	a)nie napisałeś na końcu programu . (kropki) b)brak napisu zamykającego
11	napisałeś coś w zbyt dalekiej kolumnie wciśnij klawisz END aby przekonać się jak daleko jest napis skasuj go lub napisz w mniejszej kolumnie
15	a)Nieistniejący moduł prawdopodobnie źle napisałeś nazwę modułu CRT. (np. CTR) b)musisz zmienić katalog tam gdzie masz moduły
26	niezgodność typów: do zmiennej którą zadeklarowałeś jako INTEGER stawiasz np. wyniki dzielenia lub inne obliczenia których wynik jest typu REAL. Zmień typ zmiennej w bloku VAR lub użyj zaokrąglenia ROUND(.....) w celu zamiany typu REAL na INTEGER
36	a)brak BEGIN rozpoczynającego program główny b)nie napisałeś USES lub źle napisałeś słowo USES c)nie ma słowa PROGRAM
40	przy porównywaniu w instrukcji IF napisałeś := a powinny być =
42	błędnie napisany wzór
64	napisałeś nawias kwadratowy zamiast okrągłego
85	a)brak średnika na końcu poprzedniej linii b)brak END zamykającego instrukcję złożoną
86	a)nie napisałeś BEGIN rozpoczynający program główny b)w zmiennych nie może być nawiasów ()
91	powinno być :=
100	chcesz odczytać z dysku element pliku o za dużym numerze
104	nie otworzyłeś zbioru do czytania instrukcją RESET
113	definiujesz procedurę gdy rozpoczęty został program główny słowem BEGIN
121	zadeklarowałeś tablicę jednowymiarową a odwołujesz się do tablicy dwuwymiarowej
122	brak przypisania funkcji do zmiennej
200	w programie występuje dzielenie przez zero lub uruchamiasz pascala na zaszybkim komputerze → zainstaluj patcha.

```

PROGRAM KREMS;
USES CRT;
VAR
D1,D2,P,O: ^REAL;
PROCEDURE ILE_PAMIECI_KREM;
BEGIN
  writeln('Wolne: ', MemAvail, ' max. blok: ', MaxAvail, ' bajtow. ');
end;
BEGIN
CLRSCR;
WRITELN('ILOSC PAMIECI PRZED INICJACJA ZMIENNYCH');
ILE_PAMIECI_KREM;
NEW(D1);
NEW(D2);
NEW(P);

```

```

NEW(O);
Writeln('ILOSC PAMIECI PO INICJACJI ZMIENNYCH');
ILE_PAMIECI_KREM;
Write('PODAJ PIERSZA PRZEKATNA=');
Readln(D1^);
Write('PODAJ DRUGA PRZEKATNA=');
Readln(D2^);
ILE_PAMIECI_KREM;
P^:=D1^*D2^/2;
O^:=4*SQRt((1/2*D1^)*(1/2*D1^)+(1/2*D2^)*(1/2*D2^));
Writeln('POLE=',P^:6:2);
Writeln('OBWOD=',O^:6:2);
ILE_PAMIECI_KREM;
Dispose(D1);
Dispose(D2);
Dispose(P);
Dispose(O);
Writeln('ILOSC PAMIECI PO SKASOWANIU ZMIENNYCH Z PAMIECI');
ILE_PAMIECI_KREM;
REPEAT UNTIL KEYPRESSED;
END.

```

```

PROGRAM WYWOLANIE_Z_PARAMETRAMI;
USES
  CRT,DOS;
VAR
  I,K,A,BIT:INTEGER;
BEGIN
  CLRSCR;
  IF PARAMCOUNT=0 THEN {GDY PROGRAM JEST WYWOLANY BEZ PARAMETRU}
    Writeln('PROGRAM MUSI MIEC PARAMETRY ')
  ELSE
    BEGIN
      IF PARAMSTR(1)<>'?' THEN {GDY PARAMETREM JEST ?}
        BEGIN
          IF PARAMSTR(1)='LOL' THEN
            BEGIN
              Writeln('ZNASZ HASLO');

              VAL(PARAMSTR(2),A,K); {ZAMIANA TEKSTU NA LICZBE}
              IF K=0 THEN
                BEGIN
                  Writeln('KONWERSJA POPRAWNA ');
                  BIT:=1024*8*A;
                  Writeln('WYWOLALES PROGRAM ZE ZMIENNA KILOBAJTY=',A);
                  Writeln('ILOSC BITOW TO', BIT);
                END
              ELSE
                Writeln('KONWERSJA NIEPOPRAWNA ');
            END
          ELSE
            BEGIN
              Writeln('NIE ZNASZ HASLA');
            END;
          END
        ELSE
          BEGIN
            Writeln('JEST TO PROGRAM DO ZAMIANY KILOBAJTOW NA BITY');
          END;
        END;
      REPEAT UNTIL KEYPRESSED;
    END.

```



```

Program zamiana;
uses crt;
var n:integer;
function dec_2_bin(n:integer):string;
var pomoc:string;
begin
  pomoc:="";
  repeat
    if n mod 2=0 then pomoc:='0'+pomoc;
    if n mod 2=1 then pomoc:='1'+pomoc;
    n:=n div 2;
  until n=0;
  dec_2_bin:=pomoc;
end;

```

```

Begin
  clrscr;
  write('podaj n=');
  readln(n);
  writeln('n=',n,' ',dec_2_bin(n));

```

```

  repeat until keypressed;
end.

```

```

Program dec_to_bin;
uses crt;
var
  i:integer;
  czysc:string;
  a:array[1..214] of integer;
  zbior_z_wynikami:text;
function dec_2_bin(n:integer):string;
var
  pomoc:string;
begin
  pomoc:="";
  repeat;
    if n mod 2=0 then pomoc:='0'+pomoc;
    if n mod 2=1 then pomoc:='1'+pomoc;
    n:=n div 2;
  until n=0;
  dec_2_bin:=pomoc;
end;
begin
  clrscr;
  assign(zbior_z_wynikami,'C:\uczen\tp\94\brod_los.out');
  rewrite(zbior_z_wynikami);
  writeln(zbior_z_wynikami,'Wylosowane liczby --> Brodowski');
  randomize;
  for i:=1 to 214 do
    begin
      a[i]:=random(256);
      writeln(zbior_z_wynikami,a[i]);
    end;
  close(zbior_z_wynikami);
  assign(zbior_z_wynikami,'C:\uczen\tp\94\brod_los.out');
  reset(zbior_z_wynikami);
  read(zbior_z_wynikami,czysc);
  for i:=1 to 214 do
    begin
      read(zbior_z_wynikami,a[i]);
    end;

```

```

close(zbior_z_wynikami);
assign(zbior_z_wynikami,'C:\uczen\tp\94\br_los_2.out');
rewrite(zbior_z_wynikami);
for i:=1 to 214 do
begin
  writeln(zbior_z_wynikami,a[i],' ',dec_2_bin(a[i]));
end;
close(zbior_z_wynikami);
repeat until keypressed;
end.

```

Pliki, czyli jak uchronić dane przed zgubą

Pamięć operacyjna, wykorzystywana do przechowywania danych przetwarzanych przez program, ma dwie zasadnicze wady: ograniczoną pojemność i ulotność (to poetyckie słowo oznacza po prostu, że zawartość pamięci jest tracona w chwili wyłączenia zasilania). Komputer umożliwiający przechowywanie danych wyłącznie podczas włączenia do sieci byłby raczej mało użyteczny, dlatego też projektanci sprzętu opracowali szereg urządzeń - tzw. pamięci masowych - pozwalających na trwałe przechowywanie danych. Rolę pamięci masowej w komputerze osobistym pełnią dyskietki oraz dyski twarde (oba rozwiązania wykorzystują identyczną metodę zapisu i różnią się rozwiązaniami technologicznymi). Z logicznego punktu widzenia, dane zapisywane na dyskach organizowane są w pliki, a te z kolei przechowywane są w katalogach. Całością zarządza system operacyjny, z którego usług korzystają programy użytkowe.

Rzecz jasna, również i Turbo Pascal dysponuje możliwością korzystania z plików. Z trzech dostępnych w Pascalu rodzajów plików - elementowych (jednorodnych), tekstowych i amorficznych - omówimy dwa pierwsze, mające największe zastosowanie w praktyce.

Na początek nieco teorii. Sam plik jest pewną strukturą danych zapisaną na dysku i identyfikowaną za pomocą nazwy (ściślej - ścieżki dostępu). Dane przechowywane w pliku mogą mieć reprezentację binarną (taką samą, jak w pamięci komputera) lub tekstową (taką, jaka używana jest do wprowadzania informacji z klawiatury i wyprowadzania jej na ekran monitora lub drukarkę). Reprezentacjom tym odpowiadają w Pascalu pliki elementowe oraz tekstowe.

* Pliki elementowe przechowują dane w postaci binarnej, zaś pliki tekstowe - w postaci wierszy tekstu zakończonych znakami końca wiersza. Zawartość plików elementowych jest na ogół nieczytelna dla użytkownika, natomiast treść pliku tekstowego daje się łatwo odczytać i zinterpretować. Z drugiej strony, binarna reprezentacja danych jest bardziej zwarta i oszczędna.

* Wszystkie dane przechowywane w plikach elementowych muszą być tego samego typu (prostego lub strukturalnego). Pliki tekstowe, wykorzystujące znakowe (sformatowane) reprezentacje danych, mogą być użyte do przechowywania mieszanych typów danych (np. tekstów i liczb), gdyż wszelka informacja przechowywana jest w nich w postaci tekstowej. Pliki tekstowe umożliwiają również formatowanie zapisu i korzystanie z procedur `readln` i `writeln`, które są niedostępne dla plików elementowych.

* Pliki elementowe umożliwiają tzw. dostęp swobodny - w dowolnym momencie można odwołać się do dowolnego elementu pliku. Pliki tekstowe są plikami o dostępie sekwencyjnym, co oznacza, że aby dostać się do wybranego elementu pliku, należy przeczytać wszystkie elementy znajdujące się przed nim.

Odwołania do pliku (zapisy, odczyty i inne operacje) realizowane są przez wywołanie odpowiednich funkcji systemu operacyjnego, który z kolei posługuje się liczbowymi identyfikatorami plików (korzystanie z nazw byłoby niewygodne). Również program pascalcowy nie odwołuje się do plików "bezpośrednio", lecz poprzez tak zwane zmienne plikowe, czyli złożone struktury danych reprezentujące fizyczne pliki zapisane na dysku. Ogólny schemat operacji plikowej w Pascalu obejmuje cztery etapy:

- * skojarzenie zmiennej plikowej z odpowiednim plikiem (znajdującym się na dysku lub nowo tworzonym);
- * otwarcie pliku, przygotowanie go do zapisywania lub odczytywania informacji;
- * jedna lub więcej operacji zapisu lub odczytu danych;
- * zamknięcie pliku i przerwanie skojarzenia pomiędzy zmienną plikową i plikiem.

Samą zmienną plikową deklaruje się w sposób następujący:

```

nazwa : file of typ { dla pliku elementowego }
nazwa : text { dla pliku tekstowego }

```

gdzie typ określa typ elementu składowego pliku i może być dowolnym identyfikatorem typu prostego lub strukturalnego (z wyjątkiem typu plikowego i obiektowego). Zauważ, że pojedynczym elementem pliku elementowego jest nie bajt, lecz właśnie obiekt zadanego typu (co jest dość logiczne). Dlatego też do pliku rekordów (drugi przykład poniżej) możesz wpisywać wyłącznie całe rekordy (zapisywanie lub odczytywanie pojedynczych pól jest niewykonalne), a jego długość będzie zawsze równa wielokrotności rozmiaru pojedynczego rekordu. Podobnie deklaracja pliku, którego elementami składowymi są tablice liczb całkowitych, zmusi Cię do zapisywania i odczytywania całych tablic, gdyż odwołanie się do pojedynczego elementu tablicy będzie niemożliwe. Ten sam plik można oczywiście otworzyć jako plik liczb całkowitych, co pozwoli nam na odczytywanie pojedynczych wartości. W przypadku, gdy plik przechowuje jednorodne dane, deklaracja zmiennej plikowej wykorzystuje na ogół elementy składowe odpowiedniego typu prostego. Dla baz danych, złożonych z rekordów (przechowujących dane różnych typów), jedynym wyjściem jest deklaracja pliku rekordów.

Oto kilka przykładowych deklaracji:

```
var
    Probki : file of real; { plik liczb rzeczywistych }
    KatalogNaDysku : file of Ksiazka; { plik rekordów }
    KatalogTekstowy : text; { plik tekstowy }
```

Dla potrzeb naszego programu bibliotecznego możemy wykorzystać drugi lub trzeci z powyższych przykładów. Najpopularniejszym rozwiązaniem dla baz danych (a nasz program jest właśnie prostą bazą danych) są - jak już powiedziano - pliki rekordów, umożliwiające swobodny dostęp do danych i lepsze zagospodarowanie dysku. Jeśli jednak zależy Ci na czytelności pliku z danymi, możesz wykorzystać reprezentację w postaci pliku tekstowego.

Po zadeklarowaniu odpowiedniej zmiennej plikowej można przystąpić do właściwych operacji związanych z zapisywaniem i odczytywaniem danych. Przede wszystkim musimy skojarzyć zmienną plikową z fizycznym plikiem znajdującym się na dysku. Służy do tego procedura assign:

```
assign(zmienna-plikowa, nazwa-pliku)
```

Nazwa-pliku określa tu plik, do którego chcemy się odwoływać (łącznie z ewentualną ścieżką dostępu, czyli nazwą dysku, katalogu i ewentualnych podkatalogów zawierających plik). Po wykonaniu procedury assign wszelkie odwołania do zmiennej plikowej będą dotyczyły skojarzonego z nią pliku (o nazwie którego możemy zapomnieć). Jest to dość istotne, gdyż jednym z błędów często popełnianych przez początkujących programistów jest próba odwoływania się do pliku przez podanie jego nazwy, co jest rzecz jasna nielegalne.

Przykładowe skojarzenie zmiennej plikowej z plikiem może mieć postać

```
assign(KatalogNaDysku, 'c:\biblio\dane\katalog.dat')
```

lub (lepiej)

```
assign(KatalogNaDysku, NazwaPliku)
```

W drugim przypadku nazwa pliku przekazywana jest jako zmienna, co umożliwia np. wprowadzenie jej z zewnątrz ("zaszycie" nazwy wewnątrz programu zmniejsza jego uniwersalność).

Następnym krokiem jest otwarcie pliku, czyli przygotowanie go do odczytu lub zapisu. Konieczność otwarcia (i późniejszego zamknięcia) pliku wynika z metod obsługi plików przyjętych w systemie operacyjnym, którego funkcje są zresztą w tym celu wykorzystywane. Wymiana informacji pomiędzy plikiem a programem możliwa jest dopiero po otwarciu tego ostatniego.

Dwoma podstawowymi procedurami używanymi w Pascalu do otwierania plików są reset i rewrite:

```
reset(zmienna-plikowa)
rewrite(zmienna-plikowa)
```

Procedura reset umożliwia otwarcie już istniejącego pliku, ustawiając tzw. wskaźnik plikowy na jego początek. W przypadku, gdy otwierany plik nie istnieje, wywołanie procedury reset kończy się błędem wykonania. Z kolei rewrite umożliwia otwarcie pliku niezależnie od tego, czy istniał on poprzednio: jeśli nie - tworzy ona nowy plik o danej nazwie, zaś jeśli tak - zeruje długość istniejącego pliku i ustawia wskaźnik plikowy na jego początek (czego efektem jest utracenie wszystkich danych zawartych w pliku). Warto pamiętać, że w przypadku plików tekstowych procedura reset otwiera plik wyłącznie do odczytu, zaś rewrite - wyłącznie do zapisu (nie ma zatem możliwości

mieszania odczytów i zapisów w jednym cyklu otwarcia). Zasada ta nie obowiązuje dla plików elementowych, które można odczytywać i zapisywać bez ograniczeń niezależnie od tego, czy zostały otwarte za pomocą procedury reset, czy rewrite (w tym ostatnim przypadku trzeba najpierw zapisać do pliku jakieś dane).

Sam wskaźnik plikowy jest po prostu kolejnym numerem elementu (nie bajtu!) w pliku, przy czym numeracja rozpoczyna się od zera. Każda operacja odczytu lub zapisu powoduje przesunięcie wskaźnika o wartość równą liczbie odczytanych lub zapisanych elementów, przy czym dla plików o dostępie swobodnym (elementowych) możliwe jest również jego dowolne przestawianie (nawet poza koniec pliku, choć ma to mały sens i najczęściej powoduje błędy wykonania).

Trzecią procedurą otwierającą, dostępną wyłącznie dla plików tekstowych, jest Append. Procedura ta otwiera plik do dopisywania, tj. otwiera go do zapisu nie niszcząc poprzedniej zawartości i ustawia wskaźnik plikowy na jego końcu. Umożliwia to dodawanie danych do plików tekstowych, które - jako pliki o dostępie sekwencyjnym - nie umożliwiają programowego przestawiania wskaźnika plikowego.

Do wymiany danych pomiędzy programem a plikiem służą znane nam już procedury read (odczyt) i write (zapis). Ponieważ w "standardowej" wersji obsługują one ekran monitora i klawiaturę, niezbędne jest podanie dodatkowego argumentu określającego plik, z/do którego informacja ma być odczytana lub zapisana. Argumentem tym jest właśnie nazwa odpowiedniej zmiennej plikowej:

```
read(zmienna-plikowa, lista-elementów)
write(zmienna-plikowa, lista-elementów)
```

Powyższe operacje odnoszą się zarówno do plików elementowych, jak i tekstowych. Dla tych ostatnich możliwe jest ponadto użycie procedur readln i writeln, odczytujących lub zapisujących dane wraz ze znakami końca wiersza. Ponieważ pliki elementowe przechowują wyłącznie dane określonego typu i nie mogą zawierać znaków końca wiersza, użycie procedur readln i writeln jest w ich przypadku nielegalne. Drugą istotną różnicą jest zawartość listy-elementów. W przypadku plików tekstowych lista ta może zawierać dowolne zmienne, stałe i wyrażenia (gdyż wszystkie zapisywane są w pliku w postaci tekstu), natomiast dla plików elementowych jej składnikami mogą być wyłącznie zmienne odpowiedniego typu. Dzieje się tak dlatego, że plik elementowy może zawierać wyłącznie dane jednego typu, zaś poszczególne elementy listy przekazywane są przez nazwę.

Po wykonaniu żądanych operacji zapisu i odczytu danych plik należy zamknąć. Ta bardzo ważna operacja jest ignorowana przez wielu programistów, co w efekcie prowadzi do przykrych niespodzianek w postaci zgubionych danych. U podłoża całego problemu leży tak zwane buforowanie operacji dyskowych, czyli technika polegająca na odczytywaniu i zapisywaniu danych nie pojedynczo, lecz całymi paczkami, za pośrednictwem specjalnego obszaru pamięci - tzw. bufora dyskowego. Wykorzystanie bufora pozwala na zredukowanie liczby fizycznych odczytów i zapisów na dysku, a przez to zmniejszenie jego mechanicznego obciążenia i poprawę wydajności operacji dyskowych. Ponieważ jednak podczas zapisu zawartość bufora wysyłana jest na dysk dopiero po jego zapełnieniu (lub w chwili zamknięcia pliku), przerwanie wykonywania programu może spowodować utratę danych. Również poprawne zakończenie programu powoduje co prawda automatyczne zamknięcie otwartych plików, nie opróżnia jednak buforów przechowujących nie zapisane jeszcze dane, co może spowodować ich utratę. W przypadku, gdy plik wykorzystywany jest wyłącznie do odczytu danych, niezamknięcie nie powoduje utraty informacji, co nie znaczy, że można je sobie odpuścić, bowiem lenistwo takie zwykle mści się w najmniej stosownych okolicznościach.

Pamiętaj: zamknięcie pliku jest praktycznie jedynym sposobem na bezpieczne zapisanie w nim wszystkich danych.

Na szczęście zamknięcie pliku jest bardzo proste. Realizuje je procedura close:

```
close(zmienna-plikowa)
```

Jej wywołanie ma tę samą formę dla wszystkich rodzajów plików.

Tyle teorii. Aby wprowadzić ją w życie, spróbujmy napisać zestaw procedur pozwalających na zapisanie naszego katalogu w pliku dyskowym i odczytanie go z pliku. Zgodnie z tym, co powiedzieliśmy wcześniej, do przechowywania zawartości katalogu wykorzystamy plik elementowy typu file of record.

```
procedure ZapiszNaDysku(NazwaPliku : string);
```

```
var
```

```
  f : file of Ksiazka;
```

```
  i : integer;
```

```

begin
    assign(f, NazwaPliku);    { skojarz plik ze zmienną plikową }
    rewrite(f);               { otwórz (utwórz) plik }
    for i := 1 to LbPoz do    { zapisz kolejne rekordy }
        write(f, Katalog[i]);
    close(f);                 { zamknij plik }
end;

```

Powyższa procedura ilustruje typową metodę zapisywania w pliku zawartości bazy danych. Po skojarzeniu pliku z odpowiednią zmienną i otwarciu go zapisujemy kolejne rekordy znajdujące się w tablicy (zwróć uwagę, że rekordy zapisywane są w całości, a nie polami). Po zapisaniu właściwej liczby rekordów zamykamy plik... i to wszystko. Musisz jeszcze zdawać sobie sprawę, że każde otwarcie będzie powodowało utratę poprzedniej zawartości pliku (rewrite!), ale ponieważ przed chwilą niejawnie założyliśmy, że każdorazowo zapisujemy cały katalog, jest to do przyjęcia.

Odczytanie zawartości katalogu z pliku przebiega według nieco innego schematu. Ponieważ nie wiemy, ile rekordów mamy właściwie odczytać, nie możemy zastosować pętli for. Najpopularniejszym rozwiązaniem jest czytanie kolejnych rekordów do momentu napotkania końca pliku, co wykrywane jest przez procedurę eof (ang. end-of-file - koniec pliku). Jak nietrudno się domyślić, tym razem zastosujemy pętlę while (lub repeat):

```

procedure OdczytajZDysku(NazwaPliku : string);

var
    f : file of Ksiazka;
    i : integer;

begin
    assign(f, NazwaPliku);    { skojarz plik ze zmienną plikową }
    reset(f); { otwórz plik (musi istnieć!) }
    i := 0;   { wyzeruj licznik rekordów }
    while not eof(f) do      { czytaj aż do końca pliku }
        begin
            Inc(i); { kolejny rekord }
            read(f, Katalog[i]);
        end;
    LbPoz := i; { wczytano tyle rekordów }
    close(f);
end;

```

Pozostałe operacje wykonywane w procedurze są praktycznie takie same, jedynie do otwarcia pliku wykorzystujemy tym razem procedurę reset. Zauważ, że konstrukcja procedur ZapiszNaDysku i OdczytajZDysku przewiduje przekazanie nazwy pliku jako parametru, co z kolei pozwala na ich użycie bez konieczności każdorazowego komunikowania się z użytkownikiem (nazwę można "zaszyć" w programie jako stałą). Jednym z możliwych (i często stosowanych) rozwiązań kwestii zapamiętywania danych jest każdorazowe zapisywanie całej bazy w chwili zakończenia programu i odczytywanie jej zaraz po uruchomieniu. W tym celu wystarczy na początku części operacyjnej (przed wywołaniem menu) wstawić instrukcję

```
OdczytajZDysku(PLIK_KATALOGU);
```

zaś przed samym końcem programu dopisać

```
ZapiszNaDysku(PLIK_KATALOGU);
```

przy czym stałą PLIK_KATALOGU należy wcześniej zdefiniować jako np.

```
PLIK_KATALOGU = 'Katalog.dat'
```

Metoda ta pozwala na każdorazowe zapamiętywanie treści katalogu po zakończeniu programu i jej odtwarzanie na początku kolejnej sesji bez konieczności podawania nazwy pliku (uwaga: przed pierwszym uruchomieniem programu musisz utworzyć pusty plik o nazwie KATALOG.DAT, w przeciwnym przypadku próba otwarcia nie powiedzie się). Innym sposobem jest uzupełnienie menu o polecenia zapisu i odczytu danych, co jednak wiąże się z koniecznością wywoływania odpowiednich poleceń. Jeśli wreszcie chcesz dać użytkownikowi możliwość zmiany nazwy pliku z danymi, musisz uzupełnić program o odpowiednie instrukcje wczytujące ją z klawiatury.

Do przechowywania danych można również wykorzystać plik tekstowy, jednak operacje odczytu i zapisu będą nieco bardziej skomplikowane. Wymiana danych z plikiem tekstowym odbywa się tak samo, jak z monitorem i klawiaturą, a więc poszczególne pola rekordów należy zapisywać i odczytywać indywidualnie. W zamian za to uzyskujemy możliwość łatwego obejrzenia treści pliku (gdyż zawiera on wyłącznie tekst), a także skierowania (lub odczytania) danych do (z) jednego ze standardowych urządzeń wyjścia (wejścia) obsługiwanych przez system operacyjny (np. drukarki). Oto przykład procedury zapisującej katalog do pliku tekstowego (operację odczytu możesz zrealizować w ramach ćwiczeń):

```
procedure ZapiszNaDysku(NazwaPliku : string);

var
    f : text; { tym razem plik tekstowy }
    i : integer;

begin
    assign(f, NazwaPliku);    { skojarz plik ze zmienną plikową }
    rewrite(f);               { utwórz plik }
    for i := 1 to LbPoz do    { zapisz kolejne rekordy }
        with Katalog[i] do   { wyprowadź poszczególne pola rekordu }
            begin
                writeln(f, 'Pozycja katalogu: ', i);
                writeln(f, 'Tytuł: ', Tytuł);
                writeln(f, 'Autor: ', Autor);
                if Wypozyczajacy = '' then { nikt nie wypożyczył }
                    writeln(f, 'Książka znajduje się na polce.')
                else { pole zawiera nazwisko wypożyczającego }
                    writeln(f, 'Wypozyczajacy: ', Wypozyczajacy);
                writeln(f);
            end;
        close(f); { zamknij plik }
    end;
end;
```

Sposób zapisania treści rekordu do pliku przypomina procedurę WypiszDane (zobacz poprzedni rozdział), z tym, że każda instrukcja write(ln) uzupełniona jest o specyfikację zmiennej plikowej, np.

```
writeln(f, 'Autor: ', Autor);
```

Odczytywanie danych z plików tekstowych nastręcza nieco więcej kłopotów, zwłaszcza gdy plik zawiera informacje mieszanych typów (tekst i liczby). Ceną za "luźny" format zapisu jest najczęściej konieczność tworzenia skomplikowanych procedur realizujących konwersje typów i sprawdzanie poprawności danych.

Na szczęście pliki tekstowe posiadają również zalety (jedną z nich jest właśnie czytelny format). Ponieważ standardowe urządzenia wyjściowe są w systemie DOS obsługiwane tak samo, jak pliki tekstowe, aby wyprowadzić treść katalogu na ekran (tzw. konsolę operatorską), wystarczy następujące wywołanie:

```
ZapiszNaDysku('con');
```

zaś aby wydrukować katalog na drukarce, musisz użyć wywołania

```
ZapiszNaDysku('prn');
```

gdzie con i prn są nazwami systemowych urządzeń wyjścia (konsoli i drukarki).

Na koniec wróćmy na chwilę do plików elementowych, by powiedzieć kilka słów o metodach realizowania swobodnego dostępu do danych. Bieżący element pliku (tj. element, którego dotyczyła będzie kolejna instrukcja read lub write) wskazywany jest przez wspomniany już wskaźnik plikowy. Wywołanie

```
Seek(zmienna-plikowa, numer-elementu)
```

powoduje ustawienie wskaźnika plikowego tak, by kolejna operacja odczytu lub zapisu rozpoczęła się od elementu danego numerem-elementu (elementy numerowane są od zera). Dodatkowymi funkcjami wykorzystywanymi podczas manipulowania wskaźnikiem plikowym są FilePos, zwracająca jego bieżącą wartość, oraz FileSize, zwracająca rozmiar pliku. Tak więc - zakładając, że f jest plikiem typu file of integer - wywołanie

```
Seek(f, FileSize(f));
```

ustawi wskaźnik plikowy za ostatnim elementem pliku (czyli przygotowuje plik do dopisywania), zaś konstrukcja

```
for i := 1 to 10 do
begin
    Seek(f, random(FileSize(f));
    read(f, i);
    writeln(i);
end;
```

odczyta z pliku dziesięć losowo wybranych liczb (funkcja random(k) zwraca liczbę pseudoprzypadkową z zakresu od zera do k) i wyświetli je na ekranie.

Swobodny dostęp do danych, chociaż bardzo przydatny, nie powinien być nadużywany. Wszelkie operacje na plikach są znacznie bardziej czasochłonne, niż operacje na danych umieszczonych w pamięci, zaś operacje w trybie sekwencyjnym są znacznie szybsze, niż w trybie dostępu swobodnego. Niezależnie od tych uwag musisz pamiętać, że wszelkie manipulacje na zawartości plików wiążą się z ryzykiem utraty danych w przypadku awarii systemu (np. wyłączenia zasilania). Dlatego też pliki należy zamykać bezpośrednio po wykonaniu niezbędnych czynności. Zapamiętaj

- * Do trwałego przechowywania informacji w systemie komputerowym służą pliki.
- * Turbo Pascal umożliwia obsługę plików elementowych, tekstowych oraz amorficznych.
- * Pliki elementowe składają się z elementów tego samego typu (prostego lub strukturalnego). Liczba elementów jest zawsze całkowita.
- * Pliki tekstowe zawierają dane reprezentowane w postaci wierszy tekstu. Można ich używać do przechowywania danych różnych typów.
- * Pliki elementowe umożliwiają dostęp swobodny, natomiast pliki tekstowe pozwalają wyłącznie na dostęp sekwencyjny.
- * Plik reprezentowany jest w programie przez zmienną plikową, którą należy skojarzyć z fizycznym plikiem za pomocą procedury assign.
- * Operacje na zawartości pliku muszą być poprzedzone jego otwarciem (reset lub rewrite) i zakończone zamknięciem (close).
- * Swobodny dostęp do elementów pliku umożliwia procedura Seek, ustawiająca wskaźnik plikowy na zadanej pozycji.
- * Mechanizmy obsługi plików tekstowych umożliwiają przesyłanie danych nie tylko na dysk, lecz również z i do standardowych urządzeń wejścia-wyjścia systemu operacyjnego.