**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ХАРКІВСЬКИЙ ДЕРЖАВНИЙ ПОЛІТЕХНІЧНИЙ КОЛЕДЖ**

для спеціалізації
5.123.1 «Обслуговування
комп'ютерних систем і мереж»

# Методичний посібник

**для виконання практичних робіт**
**з навчальної дисципліни**

# Програмування

**2019 р.**

Методичний посібник для виконання практичних робіт з навчальної дисципліни “Програмування” для студентів для студентів спеціальності 123 "Комп'ютерна інженерія", спеціалізації 5.123.1 "Обслуговування комп'ютерних систем і мереж".

Розробник: Ярмола О. С., викладач 1 категорії.

Методичний посібник для виконання практичних робіт затверджений на засіданні циклової комісії інформаційних технологій.
Протокол № ____ від «___» _____ 20___ року
Голова циклової комісії інформаційних технологій
_____ М.М. Бочарніков
(підпис)

Схвалено методичною радою Харківського державного політехнічного коледжу

Протокол № ____ від «___» _____ 20___ року
Голова методичної ради коледжу _____ В.О. Величко
(підпис)

# Пояснювальна записка

Методичні вказівки для виконання практичних робіт складені на підставі програми нормативної навчальної дисципліни "Програмування" для студентів третього курсу спеціальності 123 "Комп'ютерна інженерія", спеціалізації 5.123.1 «Обслуговування комп’ютерних систем і мереж».

Дані методичні вказівки присвячені вивченню тем дисципліни і містять в собі текст завдання, опис порядку виконання робіт, приклад виконання та питання для самоперевірки. Метою практичних робіт є отримання студентами практичних навичок програмування мовою Pascal.

Під час виконання практичної роботи студент повинен:

- вивчити матеріал по конспекту, підручникам та іншим посібникам;
- розробити алгоритм розв’язку задачі;
- написати програмний код;
- виконати завдання на комп’ютері згідно свого варіанту, зберегти програму;
- підготувати та оформити звіт згідно вимог;
- захистити роботу.

При виконанні практичних робіт необхідно суворо додержуватись правил техніки безпеки. Студенти допускаються до виконання практичних робіт тільки після проведення інструктажу з охорони праці при роботі в лабораторії з реєстрацією у відповідному журналі.

# Практична робота №1
# Тема: Складення програм з використанням умовних операторів.

*Мета роботи*: вивчення оператора розгалуження. Придбання навичок програмування розгалужених обчислювальних процесів.

**Теоретичні відомості.**

Правила складання алгоритму виконання студент може переглянути в лекціях чи методичних вказівках до предмету «Алгоритми та методи обчислень» (тема «Побудова алгоритмів з розгалуженням»).

**Оператор if ... then ...**

Оператор if ... then ... називається умовним оператором і має вигляд

*if expression then statement;*

де вираз expression1 є логічним. Логічне вираз приймає одне з двох можливих значень - True (істина) або False (неправда). Часто в ролі логічного виразу виступає якась умова, яка може виконуватися або ні. У першому випадку його значення - «істина», а в другому - «брехня». Програмування логічних виразів ми будемо розбирати пізніше. Якщо логічний вираз expression1 приймає значення «істина», то виконується оператор statementl. В іншому випадку виконуватися буде оператор, наступний за даними логічним оператором.

Слід зазначити, що, згідно формальним правилам мови, в умовному операторі після then допускається застосування тільки одного оператора. Але в практиці програмування частіше виникають ситуації, коли при виконанні умови в логічному вираженні expression1 слід виконати декілька операторів мови. Вирішується ця проблема застосуванням складеного оператора.

Оператори if ... then ... можна вкладати один в одного, так як конструкція

*if expression2 then statement2;*

також є оператором і може замістити оператор statementl:

*if expression1 then*
*if expression2 then statement2;*

**Оператор if ... then ... else ...**

Цей оператор є повною версією умовного оператора і має вигляд

*if expression then statementl else statement2;*

Виконується даний оператор таким чином: якщо вираз expression приймає значення «істина», то управління передається на оператор statementl, якщо ж ні, то на оператор statement2.

Оператор

```
if expression then
  if expression2 then
    statement2
  else
```

*statement;*

допускає двояку інтерпретацію. Перший варіант відповідає послідовності операторів

```
if expression then
begin
   if expression2 then
      statement2
   else
      statement1;
end;
```

Другий варіант:

```
if expression then
begin
   if expression2 then
      statement2
end
else
   statement1;
```

Компілятор Паскаля завжди вибирає перший з наведених варіантів - кожному else відповідає найближчий попередній if. Якщо потрібна реалізація другого варіанту, можна використовувати операторні дужки begin ... end. У загальному випадку, щоб чітко визначити, що чому підпорядковане, використовуйте begin ... end аналогічно круглим дужкам в арифметичних виразах.

**Індивідуальні завдання.**

Створити блок-схему та скласти програму для обчислення значення функції У (згідно варіанту) в точці, заданій користувачем. Запишіть результати виконання програми для декількох значень.

| | |
|---|---|
| 1 | $y = \begin{cases} x^{e^x} + e^2, \text{якщо } 1 \le x < 2 \\ \sin(x+5)^2 + \dfrac{x+0{,}3}{9}, \text{якщо } -1 < x < 1 \\ \sqrt[3]{1+\sqrt{\lvert x-5\rvert}}, \text{якщо } -5 \le x < -1 \end{cases}$ |
| 2 | $y = \begin{cases} ln^2 x^3 + \sqrt{x}, \text{якщо } 3 < x < 6 \\ \dfrac{x}{3 + \dfrac{x^2}{5+x}}, \text{якщо } x > 10 \\ x\, ln\, x + arctgx, \text{якщо } 6 \le x \le 10 \end{cases}$ |

| | |
|---|---|
| 3 | $y=\begin{cases}1+sin^2(x+0{,}1)\text{, якщо } -1<x<1\\ \dfrac{x}{12}+\dfrac{x}{\dfrac{x^2}{5+x}+\sqrt{\lvert x\rvert}}\text{, якщо } x\le -1\\ tgx-\text{в інших випадках}\end{cases}$ |
| 4 | $y=\begin{cases}cos^2\lvert x\rvert\text{, якщо } -2<x<-1\\ \sqrt[3]{x+sin^2(x+3)}\text{, якщо } x\le -1\\ arctg^2\dfrac{\lvert x\rvert}{2}\text{, якщо } x=-5\end{cases}$ |
| 5 | $y=\begin{cases}\sqrt[3]{x}+e^{x+9}\text{, якщо } 7<x<9\\ tg\sqrt{x}\text{, якщо } x=3\\ \ln\lvert x-9\rvert\text{, якщо } x=1 \text{ або } x=-1\end{cases}$ |
| 6 | $y=\begin{cases}\dfrac{5+\sqrt{x}}{\sqrt{100-x}}\text{, якщо } 1<x<2 \text{ або } 4<x<7\\ \lvert x\rvert\text{, якщо } x<-1\\ ln^2x^3\text{, якщо } x=0{,}75\end{cases}$ |
| 7 | $y=\begin{cases}\sqrt[4]{x}+e^{x^2}\text{, якщо } 1<x<1{,}5\\ \sqrt{\ln x}\text{, якщо } 0<x\le 1\\ \dfrac{5+\sqrt{\lvert x\rvert}}{2+2x}\text{, в інших випадках}\end{cases}$ |
| 8 | $y=\begin{cases}\dfrac{1+x^2}{\sqrt{1+x^4}}\text{, якщо } x\le 0\\ 2x+\dfrac{sin^2x}{2+x}\text{, якщо } x>1\\ 3\sin x-cos^2x\text{, якщо } 0<x\le 1\end{cases}$ |
| 9 | $y=\begin{cases}\frac{x+\frac{x}{2}}{2(x+cos^2x)}\text{, якщо } x\le 0\\ x^{\frac{1}{sinx}}\text{, якщо } x>1\\ 3-\sqrt{1+x^2}\text{, якщо } 0<x\le 1\end{cases}$ |
| 10 | $y=\begin{cases}\sqrt{\lvert x+(x+1)^3\rvert}\text{, якщо } x\le 2\\ \frac{3x^2+\sqrt{\lvert sinx\rvert}}{1+x^2}\text{, якщо } 2<x\le 0\\ 2\sqrt{\lvert 1+2x\rvert}\text{, в інших випадках}\end{cases}$ |
| 11 | $y=\begin{cases}\frac{3+sinx}{1+x^2}\text{, якщо } x\le 0\\ 2x^2cos^2x\text{, якщо } x>1\\ tg^2\frac{x}{2}+\cos(x+2)\text{, якщо } x=0{,}65\end{cases}$ |
| 12 | $y=\begin{cases}\sqrt{1+2x^3-sin^2x}\text{, якщо } x\le 0\\ \dfrac{2+x}{\sqrt[3]{2+e^{-0{,}1x}}}\text{, якщо } x>1\\ 2\cos(x+5)\text{, якщо } 0<x\le 1\end{cases}$ |
| 13 | $y=\begin{cases}\dfrac{x^3}{3+\dfrac{x^3}{5}}\text{, якщо } x\le -1\\ \sqrt{1+x^2}\text{, якщо } -1\le x<0\\ \dfrac{1+x}{1+\sqrt[3]{1+e^{-0{,}2x}}}\text{, якщо } x=5 \text{ або } x=10\end{cases}$ |

| | |
|---|---|
| 14 | $y=\begin{cases}\sqrt[3]{1+\sqrt{\lvert x-10\rvert}}, \text{якщо } x\le -2\\ tg^2x+1 \text{ , якщо } x>1\\ 1+(x+\cos(x-1)) - \text{в інших випадках}\end{cases}$ |
| 15 | $y=\begin{cases}\sqrt{1+\lvert x\rvert} \text{ , якщо } x\le 0\\ \dfrac{1+3x}{2+\sqrt[3]{1+x}} \text{ , якщо } x>1\\ arctg\dfrac{3x}{2} \text{ , якщо } x=0{,}33\end{cases}$ |
| 16 | $y=\begin{cases}\dfrac{\sqrt{1+\lvert x\rvert}}{2+\lvert x\rvert}, \text{якщо } x\le 0\\ \dfrac{1+x}{2+cos^3x} \text{ , якщо } x>1\\ 4\sqrt{1+x^3}, \text{в інших випадках}\end{cases}$ |
| 17 | $y=\begin{cases}\sqrt[3]{1+x^2} \text{ , якщо } x\le 0\\ sin^2x+\dfrac{1+x}{1+cos^2x} \text{ , якщо } x>5\\ tg^2x+ctgx \text{ , якщо } x=2 \text{ або } x=3\end{cases}$ |
| 18 | $y=\begin{cases}2^{-x}\sqrt{\lvert x\rvert} \text{ , якщо } x<0\\ \sqrt{e^{x+1}+sinx} \text{ , якщо } x>1\\ \ln x^2 - \text{в інших випадках}\end{cases}$ |
| 19 | $y=\begin{cases}cos^3(x+3) \text{ , якщо } x<0\\ x+arctg \text{ , якщо } x>1\\ \frac{x^2}{x+\frac{x}{x^3+1}} \text{ , якщо } x=0{,}5\end{cases}$ |
| 20 | $y=\begin{cases}1+\dfrac{x^2}{2}+\dfrac{x^3}{4} \text{ , якщо } x<0\\ xarctgx \text{ , якщо } x>5\\ \dfrac{\ln(x+0{,}1)}{\sin(x+0{,}2)} \text{ , якщо } 1<x<1{,}5\end{cases}$ |
| 21 | $y=\begin{cases}sin^2(x+5), \text{ якщо } x<0\\ \sqrt[3]{x}+e^x, \text{ якщо } x>1\\ \dfrac{cos^2x+sin^2x}{ctgx^2} \text{ , в інших випадках}\end{cases}$ |
| 22 | $y=\begin{cases}\sqrt[3]{\dfrac{x}{100}}+sin^2x^2, \text{ якщо } x<0\\ \lvert 100-x\rvert cosx, \text{ якщо } x>1\\ e^{x+1,1}+\sin x \text{ , якщо } x=1 \text{ або } x=0{,}22\end{cases}$ |
| 23 | $y=\begin{cases}\sqrt[3]{1+\sqrt{\lvert x-10\rvert}}, \text{ якщо } x\le -2\\ tg^2\,x+1, \text{ якщо } x>1\\ 1+(x+\cos(x-1)) - \text{в інших випадках}\end{cases}$ |
| 24 | $y=\begin{cases}\sqrt{1+\lvert x\rvert}, \text{ якщо } x\le 0\\ \dfrac{1+3x}{2+\sqrt[3]{1+x}}, \text{ якщо } x>1\\ arctg\dfrac{3x}{2} \text{ , якщо } x=0{,}33\end{cases}$ |

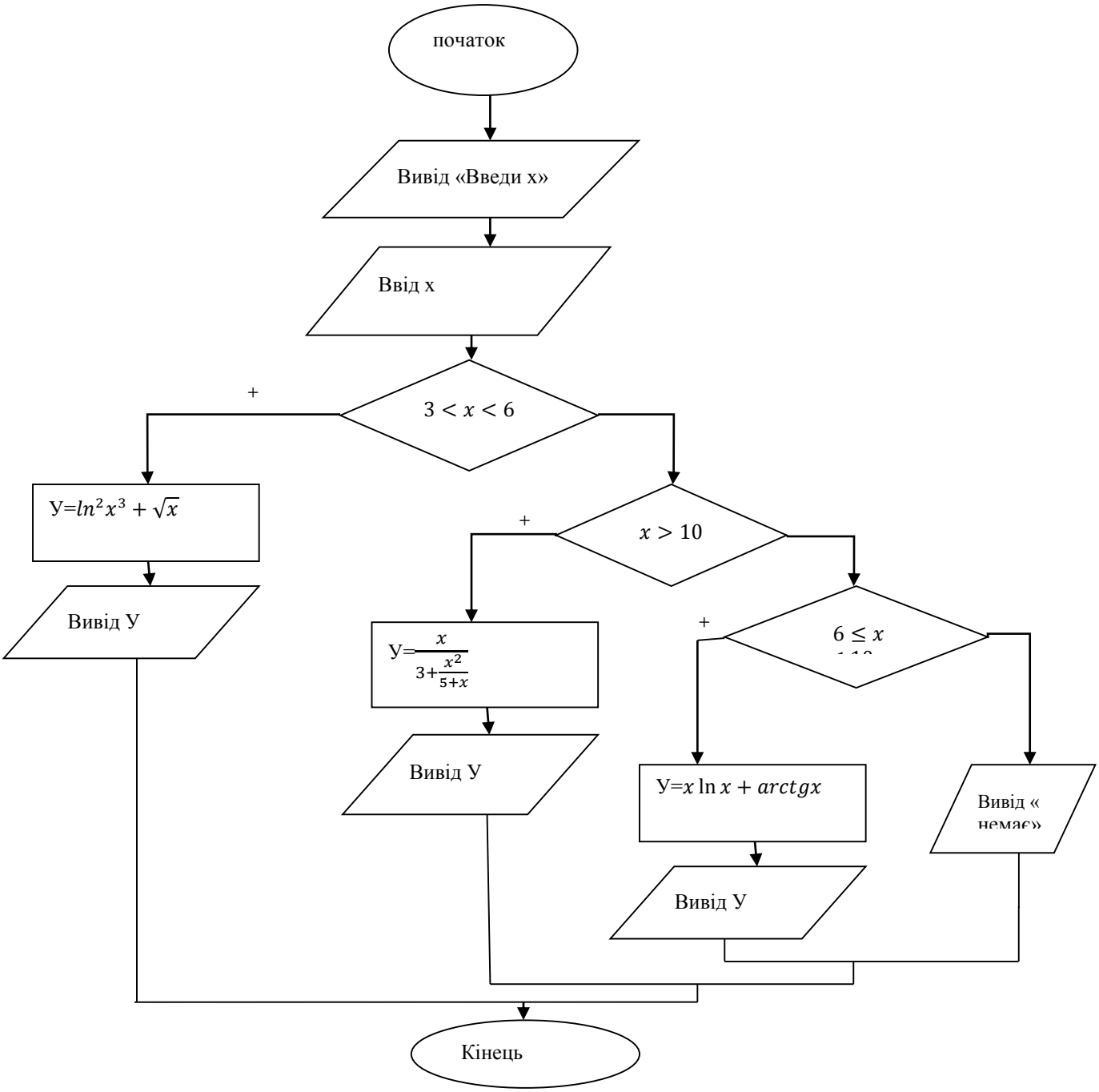
| | |
|---|---|
| 25 | $y=\begin{cases}\dfrac{\sqrt{1+\lvert x\rvert}}{2+\lvert x\rvert}\ , & \text{якщо } x\le 0\\ \dfrac{1+x}{2+cos^3x}\ , & \text{якщо } x>1\\ 4\sqrt{1+x^3}\ - \text{в інших випадках}\end{cases}$ |
| 26 | $y=\begin{cases}\dfrac{3+\sin x}{1+x^2}\ , & \text{якщо } x\le 0\\ 2x^2cos^2x\ , & \text{якщо } x>1\\ tg^2\dfrac{x}{2}+\cos(x+2)\,, & \text{якщо } x=0{,}65\end{cases}$ |
| 27 | $y=\begin{cases}\sqrt{1+2x^3-sin^2x}\ , & \text{якщо } x\le 0\\ \dfrac{2+x}{\sqrt[3]{2+e^{-0,1x}}}\ , & \text{якщо } x>1\\ 2\cos(x+5), & \text{якщо } 0<x\le 1\end{cases}$ |
| 28 | $y=\begin{cases}\dfrac{x^3}{3+\dfrac{x^3}{5}}\ , & \text{якщо } x\le -1\\ \sqrt{1+x^2}\,, & \text{якщо } 0<x\le 1\ x=5 \text{ або } x=10\\ \dfrac{1+x}{1+\sqrt[3]{1+e^{-0,2x}}}, & \text{якщо } x=5 \text{ або } x=10\end{cases}$ |
| 29 | $y=\begin{cases}\sqrt[3]{1+\sqrt{\lvert x-10\rvert}}\ , & \text{якщо } x\le -2\\ tg^2x+1\,, & \text{якщо } x>1\\ 1+(x+\cos(x-1))\ - & \text{в інших випадках}\end{cases}$ |
| 30 | $y=\begin{cases}ln^2x^3+\sqrt{x}\,, & \text{якщо } 3<x<6\\ \dfrac{x}{3+\dfrac{x^2}{5+x}}\ , & \text{якщо } x>10\\ x\ln x+arctgx\ , & \text{якщо } 6\le x\le 10\end{cases}$ |

## Контрольні питання.

1. Які обчислення називаються розгалуженими?
2. Які типи операторів розгалуження ви знаєте? Охарактеризуйте кожен з типів та наведіть приклади застосування.
3. Чи в конструкції if оператор else являється обов'язковим? Вкажіть, в яких випадках оператор else є необов'язковим?

**Приклад виконання роботи (варіант 30).**

Блок-схема:



Лістинг програми:

```
Program LR1;
Var x,y: real;
Begin
Writeln('введите значение переменной x');
Readln(x);
If (3<x) and (x<6) then
        Begin
        Y:=sqr(ln(x*x))+ sqrt(x);
        Writeln('при x=', x, ' y=', y);
        End Else
        If x>10 then
                Begin
                Y:=x/(3+(x*x)/(5+x));
                Writeln('при x=', x, ' y=', y);
                End   Else
                If (x<=10) and (6<=x) then
                        Begin
                        Y:=x*ln(x)+arctan(x);
                        Writeln('при x=', x, ' y=', y);
                        End
                Else Writeln('для этого значения x функция не определена');
End.
```

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм виконання.
- Навести програму та кінцевий результат обчислення.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №2

# Тема: Складення програм з використанням операторів циклу

*Мета роботи*: вивчення оператора циклу. Придбання навичок програмування циклічних обчислювальних процесів.

**Теоретичні відомості.**

Циклічними називаються обчислювальні процеси, в яких неодноразово виконуються одні й ті ж дії, але з різними даними. Тіло циклу складається з операторів, що повторюються у програмі. Для організації циклу необхідно задати початкове значення змінної, яка буде змінюватися у циклі, її кінцевого значення та крок її зміни. Треба контролювати значення цієї змінної для перевірки умови виходу з циклу. Умовою може бути: перевищення параметром циклу кінцевого значення, виконання заданого числа повторень, досягнення заданої точності обчислення.

Цикли бувають арифметичні та ітеративні. В арифметичних циклах число повторень визначається на основі зміни параметра циклу; в ітеративних циклах - цикл повторюється доти, доки не буде виконана умова виходу з циклу.

В мові Паскаль є три спеціальні оператори організації циклів: WHILE, REPEAT та FOR.

У Паскалі передбачено три різновиди операторів циклу: цикл із передумовою, цикл з післяумовою, цикл із лічильником (із покроковою зміною аргументу). Також реалізована робота із вкладеними циклами. Вкладені цик-ли — циклічні процеси, що допускають укладеність одних циклів в інші.

Цикл із передумовою (або цикл-«поки») — це цикл, у якому тіло циклу виконується тільки у разі виконання умови, заданої перед тілом циклу. Якщо умова стає невірною, то робота циклу припиняється і керування передається оператору, наступному за оператором циклу.

На мові Паскаль оператор циклу з перед-умовою ще називається «циклом While-Do».

*WHILE <умова> DO <оператор>;*

Цикл із післяумовою (або цикл-«до») — це цикл, у якому тіло циклу виконується доти, поки умова, задана після тіла циклу, не стане правильною. Якщо умова стає правильною, то робота циклу припиняється й управління передається оператору, наступному за оператором циклу.

На мові Паскаль оператор циклу з після-умовою ще називається «цикл

Repeat-Until».

*REPEAT <оператор> UNTIL <умова>;*

Цикл із лічильником (із покроковою зміною аргумен-ту) — це цикл, у якому тіло циклу виконується заздалегідь відому кількість разів. У різних алгоритмічних мовах реалізація цього циклу може передбачати використання аргументів різних типів, зміну аргументу на різний крок, діапазон зміни аргументу і т. д.

Цикл із лічильником аргументу реалізовується таким чином:

1) аргументу надається початкове значення;

2) якщо значення входить у заданий діапазон, то виконується тіло циклу;

3) аргумент змінюється на заданий крок; виконується 2);

4) якщо значення не входить у заданий діапазон, то виконання циклу припиняється і керування передається оператору, наступному за оператором циклу.

У мові Паскаль реалізовано два оператори циклу з покроковою зміною аргументу: «цикл For-To» і «цикл For-DownTo».

Загальний вигляд оператора циклу з параметром (з лічильником):

*for <параметр циклу>:=N1 to N2 do <тіло циклу >;*

де N1 та N2 - початкове та кінцеве значення параметра циклу, тіло циклу може бути простим або складеним оператором. <Параметр циклу > ще називають лічильником циклу. Оператор for забезпечує виконання тіла циклу до тих пір, поки не будуть перебрані всі значення параметра циклу від початкового до кінцевого. Параметр циклу, його початкове та кінцеве значення повинні бути одного і того ж скалярного типу. При цьому можливий будь який стандартний тип, крім real. Якщо N1 та N2 цілі числа, а параметр циклу - цілочисельна змінна, то крок завжди рівний одиниці.

Для розв'язування переважної більшості типових задач не має значення, який цикл застосовувати: while чи repeat. Треба лише пам'ятати, що умови (логічні вирази) в цих командах є протилежні: у команді while логічний вираз описує умову продовження обчислень у циклі, а в команді repeat – умову виходу з циклу.

**Індивідуальні завдання.**

Створити блок-схему та скласти програму для обчислення значення функції У (згідно варіанту) в точці, заданій користувачем. Виконайте завдання всіма можливими способами. Запишіть результати виконання програми.

| № варіанта | Вид функції | Первинні данні | | | | |
|---|---|---|---|---|---|---|
| | | a | b | $X_н$ | $X_к$ | h |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | $y=\frac{arctgbx}{1+sin^2 x}$ | - | 0,75 | 1,35 | 6,5 | 0,8 |
| 2 | $y=\sqrt[5]{\frac{a+bx}{ln^2 x}}$ | 19,6 | 7,8 | 14,6 | 34,8 | 6 |
| 3 | $y=\frac{a\,ln^2 x}{b+\sqrt{x}}$ | 1,38 | -1,2 | 60 | 100 | 10 |
| 4 | $y=\frac{sin^2 x}{\sqrt{x}+bx}$ | - | 1,68 | 1,2 | 2,4 | 0,2 |
| 5 | $y=\frac{ln^2 (x-b)}{a\sqrt{x}}$ | 0,36 | 5,5 | 10 | 50 | 6 |
| 6 | $y=\frac{e^{xa}+b}{1+cos^2 x}$ | 0,9 | 1,85 | 0 | 1,2 | 0,15 |
| 7 | $y=\frac{a+\sqrt[3]{x}}{sin^2 bx}$ | 1,24 | 0,67 | 10,2 | 12,4 | 0,43 |
| 8 | $y=\frac{a\sqrt{x}-bx}{ln^2 x}$ | 2,8 | 0,45 | 40 | 60 | 4,5 |
| 9 | $y=\frac{\sqrt{a}x-b}{lg^3 x}$ | 20,2 | 7,65 | 3,5 | 4 | 0,1 |
| 10 | $y=e^{-x^2}\frac{a+bx}{sin\,x}$ | 4,6 | 2,5 | 0,75 | 1,8 | 0,3 |
| 11 | $y=\frac{tg^2 ax-b}{e^{ax}}$ | 0,55 | 0,78 | 4,2 | 5,8 | 0,25 |
| 12 | $y=\frac{arctgbx}{1+\sqrt[5]{ax}}$ | 7,38 | 0,3 | 9 | 12 | 0,35 |
| 13 | $y=\frac{sin^3 ax}{ax+b}$ | 0,28 | 1,35 | 1,2 | 7,5 | 0,5 |
| 14 | $y=\frac{e^{-bx}}{b+cos^3 ax}$ | 0,9 | 0,66 | 2,3 | 8,9 | 1,3 |
| 15 | $y=\frac{ln^2 \sqrt{x}}{a\sqrt{x}}$ | 0,85 | - | 17,2 | 24,6 | 2 |
| 16 | $y=\frac{arctg(a^3 x)}{\sqrt{a^3+x^3}}$ | 1,16 | - | 0,25 | 1,28 | 0,33 |
| 17 | $y=\frac{1+\sqrt{bx}}{sin^2 ax}$ | 0,4 | 10,8 | 0,84 | 1,25 | 0,15 |
| 18 | $y=\frac{a-e^{bx}}{ln^2 x}$ | 1,28 | 0,03 | 12,6 | 34,9 | 7,6 |
| 19 | $y=\frac{(a+bx)^{25}}{1+cos^3 ax}$ | 0,25 | 0,68 | 11,6 | 15,8 | 0,6 |
| 20 | $y=\frac{b+sin^2 ax}{e^{-x12}}$ | 1,6 | 1,24 | 0,2 | 1,4 | 0,35 |
| 21 | $y=\frac{sin^2 x-a}{bx}$ | 1,8 | 0,34 | 6,44 | 9,1 | 0,25 |
| 22 | $y=\frac{atg^2 x}{b+0,7x}$ | 0,44 | 2,28 | 6,5 | 7,3 | 0,12 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **23** | $y=\frac{ln(a^2-x)}{b\,sin^2\,x}$ | **3,2** | **0,45** | **0,6** | **1,5** | **0,2** |
| **24** | $y=\frac{a-\sqrt{bx}}{1+cos^2\,x}$ | **17,6** | **10,4** | **1,9** | **3,8** | **0,3** |
| **25** | $y=\frac{ln^2\lvert x+a\rvert}{(x+a)^2}$ | **8,24** | **-** | **14,9** | **24,8** | **1,5** |
| **26** | $y=\frac{\sqrt{alnx}}{1+tg^2\,bx^2}$ | **7,32** | **0,05** | **13,3** | **14,5** | **0,08** |
| **27** | $y=\frac{1+tg^2\,x/a}{b+e^{x/a}}$ | **4,1** | **0,05** | **1,25** | **3** | **0,3** |
| **28** | $y=\frac{e^{ax}+a^{e^x}}{\sqrt{1+sin^2\,x}}$ | **2** | **-** | **0,6** | **0,02** | **0,05** |
| **29** | $y=\frac{\sqrt{ax+b}}{ln^2\,x}$ | **1,35** | **0,98** | **7,5** | **26,6** | **4,2** |
| **30** | $y=\frac{sin^2(b^2+x^2)}{\sqrt[3]{b^2+x^2}}$ | **-** | **2,5** | **1,28** | **5,34** | **0,4** |

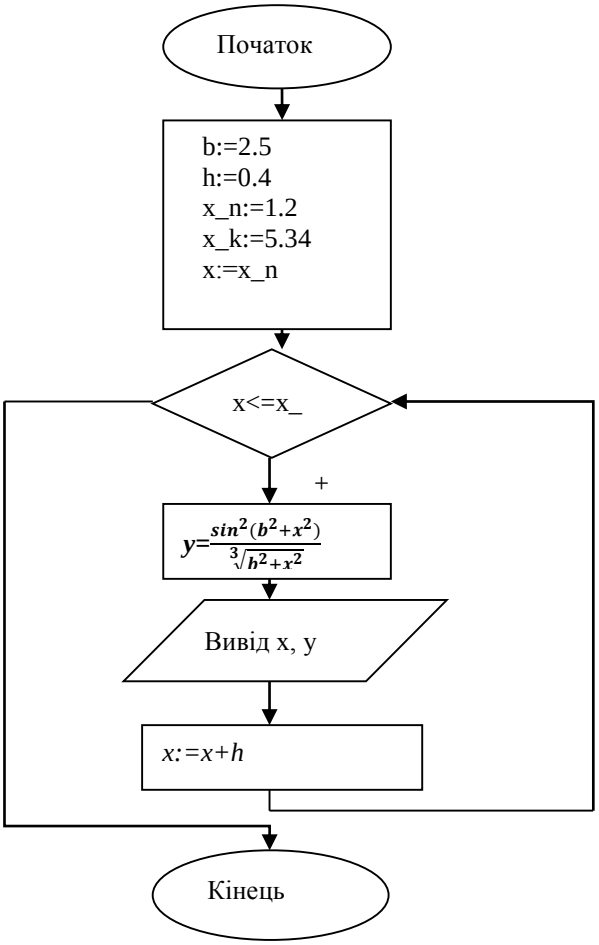
## Контрольні питання.

1. Які обчислення називаються циклічними?
2. Які типи операторів циклу ви знаєте? Охарактеризуйте кожен з типів та наведіть приклади застосування.

**Приклад виконання роботи (варіант 30).**

Виконаємо завдання за допомогою циклічного оператора while.

Блок-схема:



Лістинг програми:

```
Program LR2;
Var x, y: real;
Const b=2.5;
h=0.4;
x_n=1.28;
x_k=5.34;
begin
x:=x_n;
while x<=x_k do
  begin
  y:= (sqr(sin(b*b+x*x)))/(exp((1/3)*ln(b*b+x*x)));
  writeln ('x=', x:3:2, '  y=', y:3:2);
  x:=x+h;
  end;
end.
```

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритми виконання.
- Навести програми та кінцевий результат обчислення.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №3

# Тема: Ініціалізація масивів та робота з пам’яттю

*Мета роботи*: вивчення методів роботи з одномірними масивами. Придбання навичок завдання масивів та обробки даних.

**Теоретичні відомості (для практичних робіт №3 та №4).**

Масивом називається скінченна послідовність змінних одного типу, які мають однакове ім'я та різняться порядковим номером.

Індексом називається порядковий номер елемента масиву.

Як звернутися до елементів цього масиву? Для цього необхідно вказати індекс.

Наприклад,

T[2], T[5], T[i], T[i + j].

Масиви відносяться до структур з так званим прямим або довільним доступом: щоб визначити окремий елемент масиву, достатньо вказати його індекс.

Тепер зрозуміло, як у циклі перебирати різні значення елементів масиву: для цього достатньо змінювати їх індекси. А закон зміни індексів дуже простий - кожне наступне значення більше попереднього на одиницю. Дуже зручна закономірність!

Оскільки у мові Pascal усе з чим ми працюємо потрібно оголошувати, то масиви також потрібно оголосити. Це можна зробити кількома способами:

у полі const

*const <ім'я змінної>=array[1 .. <кількість елементів>] of <тип> = (1,2,3, ... <значення>);*

Зверніть увагу на те, що в розділі констант стоїть символ «=», а не символ «:=».Використовуючи в програмі константи, необхідно врахувати таку особливість: їх значення не можна змінювати під час виконання програми. Тобто ідентифікаторам, що визначають константи, не можна присвоювати ніяких нових значень. їх можна використовувати лише для обчислення інших значень.

у полі type

*type <ім'я типу>=array[1 .. <кількість елементів>] of <тип>;*

*var <ім'я змінної> : <ім'я типу>;*

у полі var

*var <ім'я змінної> : array[1 .. <кількість елементів>] of <тип>;*

Приклад:

*type Mas = array[1 .. 5] of integer;*

*var a : Mas;*

Масиви бувають одновимірними (у вигляді послідовності чисел), двовимірними (у вигляді таблиць чисел розміром m x n) і багатовимірними (3-,4-вимірні і т.д. 3-вімірні - це об'ємний простір з комірками, а 4-вимірні і більше - це

фантастично-абстрактні поняття).

Масив називається одновимірним, якщо для задання місцеположення елемента в масиві необхідно вказати значення лише одного індексу.

Масив називається двовимірним, якщо для задання місцеположення елемента в масиві необхідно вказати значення двох індексів.

Запам'ятайте, що у двовимірних масивах перший індекс завжди вказує на номер рядка, а другий - на номер стовпчика в цьому рядку!

Розмірність масивів у Pascal необмежена, вона визначається лише об'ємом пам'яті вашого комп'ютера.

Загальний вигляд опису масивів:

*<ім'я змінної>: array [<межі зміни індексів>] of <тип>.*

Наприклад,

varA: array[1..10] of real;B: array[1..100,1..100] of byte;C: array[1..100] of array[1..100] of byte.

Зауваження.

По-перше, межі індексів завжди вказуються через два символи «.».

По-друге, при розподілі пам'яті в описовій частині програми під масив буде зарезервовано стільки місця, скільки передбачає вказана кількість елементів масиву. Тому при виконанні програми ви можете використовувати кількість елементів не більшу, ніж описана в розділі змінних.

По-третє, межі зміни індексів повинні бути сталими величинами, а не змінними, інакше невідомо буде, скільки місця необхідно відвести в пам'яті під такий масив.

Якщо вас не цікавлять конкретні дані, які будуть надані елементам масиву під час виконання програми, то скористайтеся можливостями стандартної процедури Pascal Randomize та функції Random (n), що генерують випадкові числа. Параметр n (типу Word) у процедурі Random визначає праву межу інтервалу, в якому будуть визначатися випадкові числа (ліва межа завжди 0). Функція Random може задаватися і без параметра. У цьому разі вона генеруватиме те дійсне число в діапазоні [0; 1). А для того щоб випадкові числа в програмі з кожним її наступним запуском не повторювалися (хоча вони і випадкові, але послідовність цих чисел постійна), то скористайтеся процедурою Randomize, яка встановить початок відрахунку випадкових чисел залежно від поточного стану системного годинника вашого комп'ютера.

Фрагмент програми, що використовує випадкові числа, може виглядати так:

```
randomize;
for i := 1 to n doa[i] := random (100);
```

Одновимірні масиви

Введення масиву з клавіатури

```
for i:=1 to n do readln(a[i]);
```

тут i - параметр, n - кількість елементів у масиві, a - одновимірний масив

Друк масиву на екран

```
for i:=1 to n do writeln(a[i]);
```

Пошук мінімального/максимального елеманту

Припускаємо, що це перший і переглядаємо масив. Якщо зустрінемо більший (чи менший) за нього елемент, то цей елемент стає максимальним (мінімальним). Розглянемо пошук максимального:

```
max:=a[1];
for i:=1 to n do
  if a[i]>max then max:=a[i];
```

Сортування за зростанням/спаданням

Переглянемо масив n разів. Кожного разу розглядаємо всі елементи від 1 до n-1. Якщо елемент більший (сортування за зростанням) за нступний за ним, то міняємо їх місцями.

```
fo j:=1 to n do        {переглядаємо масив n разів}
for i:=1 to n-1 do     {переглядаємо кожного разу елементи від 1 до n-1 (бо можемо поміняти місцями a[n-1] і a[n] - а у програмі буде a[i] та a[i+1] при i=n-1)}
   if a[i]>a[i+1] then
     begin
        b:=a[i];       {зберігаємо значення a[i] в іншу змінну, тип якої такий як і елементів масиву}
        a[i]:=a[i+1];  {власне міняємо місцями елементи - записуємо менше значення на місце більшого}
        a[i+1]:=b;     {"витягуємо" з пам'яті значення a[i], більше, і записуємо на нове місце}
     end;
```

Такий метод сортування називається бульбашковим (або "Метод бульбашки").

Двовимірні масиви (n x m)

Введення масиву з клавіатури

```
for i:=1 to n do       {перебір n рядків}
  for j:=1 to m do     {перебір m стовпців}
    readln(a[i,j]);        {власне ввід кожного елементу}
```

Вивід масиву на екран

Щоб вивести двовимірний масив на екран у вигляді таблиці роблять наступне:

```
for:=1 to n do     {перебір рядків}
  begin
    for j:=1 to m do write(a[i,j],' ');      {вивід кожного рядка}
    writeln;                                 {перехід на новий рядок}
  end;
```

**Індивідуальні завдання**

Числові значення до лабораторної роботи обрати з ряду числових значень: -0.1, 100, -99.1, 2.22, 14.33, -10.07, 0.033, 4.15, -15.01, 11, -31.01, -70.1, 55.6, 30.15, 12, -0.433, 0, 16.17, 0, -19.61.

Створити алгоритм виконання та програму згідно варіанту.

1. Знайти суму елементів масиву.
2. Знайти добуток елементів масиву.
3. Знайти найбільший елемент масиву.
4. Знайти найменший елемент масиву.
5. Знайти кількість входжень в масив числа 2.
6. Знайти номер елемента масиву, що дорівнює 1.
7. Знайти номер першого входження елемента масиву, що дорівнює 2.
8. Знайти середнє значення елементів масиву.
9. Знайти кількість елементів масиву, котрі менші числа 3.
10. Знайти кількість елементів масиву, котрі більші числа 2.
11. Знайти квадрат найменшого значення з елементів масиву.
12. Знайти корінь квадратний корінь найбільшого значення з елементів масиву.
13. Знайти суму елементів масиву з парними номерами.
14. Знайти добуток елементів масиву з парними номерами.
15. Знайти суму елементів масиву з непарними номерами.
16. Знайти добуток елементів масиву з непарними номерами.
17. Знайти суму додатних елементів масиву.
18. Знайти суму від'ємних елементів масиву.
19. Знайти добуток додатних елементів масиву.
20. Знайти добуток від'ємних елементів масиву
21. Вивести на екран в рядок номера елементів масиву, які більші числа 2.
22. Вивести на екран в стовпчик номера елементів масиву, які менші числа 6.
23. Вивести на екран в рядок елементи масиву, котрі менші числа 3.
24. Вивести на екран в стовпчик елементи масиву, які більші числа 1.
25. Знайти номера елементів масиву, що дорівнюють 2.
26. Знайти кількість додатних елементів масиву.
27. Знайти кількість від'ємних елементів масиву.
28. Знайти добуток додатних елементів масиву з парними номерами.
29. Знайти суму від'ємних елементів масиву з непарними номерами.
30. Перевірити, чи є серед елементів масиву число 0.

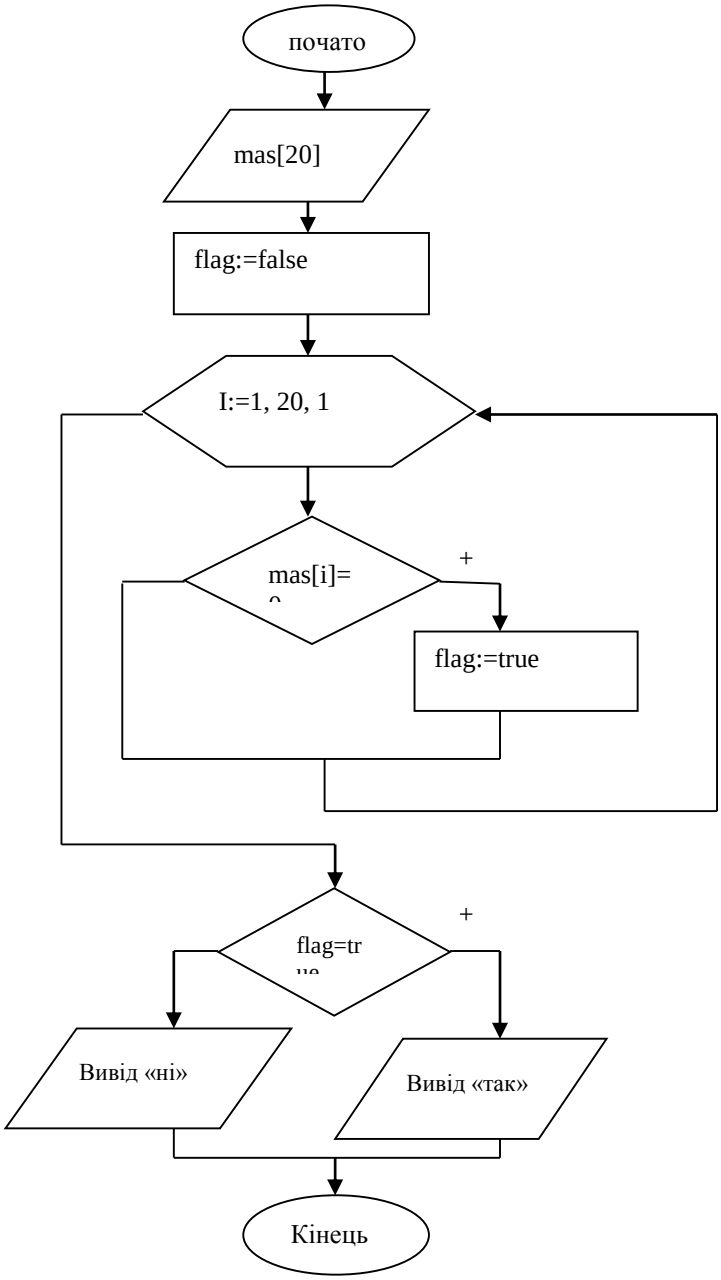
**Контрольні питання.**

1. Які обчислення називаються циклічними?
2. Які типи операторів циклу ви знаєте? Охарактеризуйте кожен з типів та наведіть приклади застосування.

**Приклад виконання роботи (варіант 30).**

Студент (на вибір) елементи масиву може вводити під час виконання програми вручну або задати всі числові значення на початку програми.

Блок-схема:




Лістинг програми:

Program LR3;

*Const mas: array [1..20] of real = (-0.1, 100, -99.1, 2.22, 14.33, -10.07, 0.033, 4.15, -15.01, 11, -31.01, -70.1, 55.6, 30.15, 12, -0.433, 0, 16.17, 0, -19.61);*

*Var I: integer;*

*Flag: boolean;*

*Begin*

*Flag:=false;*

*For I:=1 to 20 do*

```
        If mas[i]=0 then Flag:=false;
If Flag then writeln('в масиві є нульовий елемент')
Else writeln('в масиві немає нульового елементу');
End.
```

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритми виконання.
- Навести програму та кінцевий результат обчислення.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №4

# Тема: Робота з двовимірними масивами

*Мета роботи*: вивчення методів роботи з двомірними масивами. Придбання навичок завдання масивів та обробки даних.

**Індивідуальні завдання**

Числові значення задаваної матриці

| Номер рядка | | Номер стовпчика | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | 1,2 | 0,8 | -1,3 | -0,8 | 8 |
| 2 | 4,5 | 2,3 | -5,1 | 0 | 8,1 |
| 3 | 2,3 | 2,6 | 10,5 | 3,5 | 10 |
| 4 | 9,5 | -3,8 | -3,3 | 3,7 | -7,2 |
| 5 | 1,7 | 6,8 | 1,7 | 1,5 | 1,8 |

Відлік кількості числових значень треба починати з першого елемента таблиці.
Створити алгоритм виконання та програму згідно варіанту.

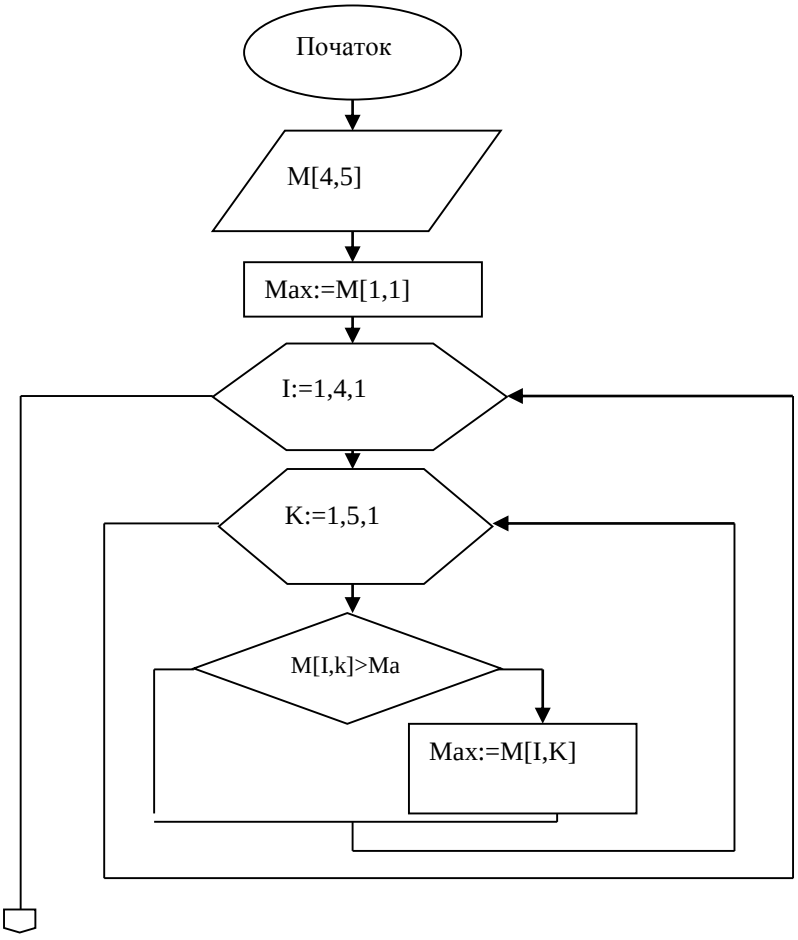
Варіанти завдання до лабораторної роботи №4

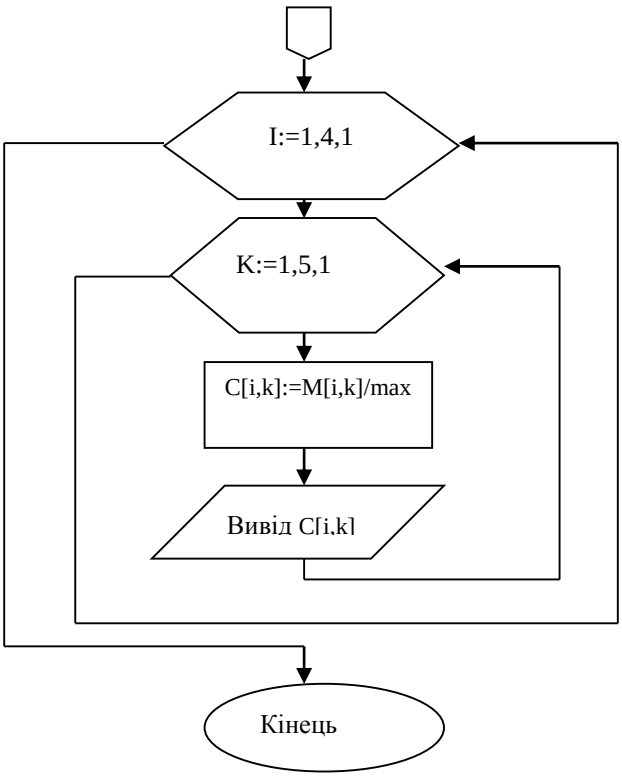
| Номер варіанта | Зміст завдання |
|---|---|
| 1 | 2 |
| 1 | Визначити суму від'ємних елементів у кожному стовпчику. Матриця A(3,4) |
| 2 | Обчислити середнє арифметичне значення додатних елементів в кожному рядку с парним номером . Матриця A (4,3) |
| 3 | Визначити кількість та суму від'ємних елементів у кожному стовпчику. Матриця A(3,4) |
| 4 | Знайти мінімальне значення на боковій діагоналі масиву. Матриця B(4,4) |
| 5 | Обчислити середнє арифметичне значення додатних елементів розташованих під головною діагоналлю масиву. Матриця B(4,4) |
| 6 | Знайти максимальне додатне значення на головній діагоналі масиву. Матриця B(4,4) |
| 7 | Обчислити добуток додатних елементів, розташованих під головною діагоналлю масиву. Матриця B(4,4) |
| 8 | Визначити число елементів, значення яких менше, ніж середнє арифметичне. Матриця A(3,4) |
| 9 | Знайти мінімальне значення на боковій діагоналі масиву. Матриця B(5,5) |
| 10 | Знайти максимальне значення на боковій діагоналі масиву. Матриця B(4,4) |
| 11 | Визначити число елементів кожного рядка, значення яких більше, ніж середнє арифметичне в цьому ж рядку. Матриця C(5,3) |
| 12 | Знайти мінімальне значення на головній діагоналі масиву. Матриця B(5,5) |
| 13 | Обчислити середнє арифметичне значення додатних елементів у кожному рядку. Матриця B(5,3) |
| 14 | Обчислити середнє арифметичне значення від'ємних елементів у кожному рядку. Матриця B(5,5) |

| | |
|---|---|
| 15 | Обчислити середнє арифметичне значення додатних елементів, розташованих над головною діагоналлю масиву. Матриця B(5,5) |
| 16 | Обчислити й надрукувати суму кожного рядка. Матриця A(5,3) |
| 17 | Обчислити й надрукувати суму кожного стовпця. Матриця A(5,4) |
| 18 | Знайти й надрукувати середнє значення кожного рядка. Матриця B(5,4) |
| 19 | Знайти різницю між мінімальним і максимальним значеннями елементів на боковій діагоналі масиву. Матриця A(5,4) |
| 20 | Знайти різницю між мінімальним і максимальним значеннями елементів масиву. Матриця A(5,4) |
| 21 | Підрахувати кількість додатних та від'ємних елементів масиву. Матриця B(4,5) |
| 22 | Обчислити й надрукувати добуток у непарних рядках. Матриця B(5,4) |
| 23 | Обчислити й надрукувати кількість нульових елементів у парних рядках. Матриця B(5,4) |
| 24 | Обчислити й надрукувати кількість нульових елементів у парних рядках.Матриця B(5,4) |
| 25 | Знайти різницю між середнім арифметичним значенням додатних елементів у парних рядках та від'ємних елементів у непарних рядках. Матриця B(5,5) |
| 26 | Сформувати матрицю, яку можна отримати з первинної діленням її від'ємних елементів на суму додатних. Матриця A(3,4) |
| 27 | Обчислити та надрукувати елементи матриці C, кожен з яких дорівнює додатку відповідних елементів заданих матриць A та B. Матриця A(3,4) береться з таблиці, матриця B(3,4) вводиться користувачем |
| 28 | Обчислити добуток усіх елементів масиву, виключаючи нульові елементи. Матриця A(5,3) |
| 29 | Обчислити суму усіх елементів масиву. Матриця A(5,4) |
| 30 | Сформувати матрицю, яку можна отримати з первинної діленням її елементів на максимальний елемент масиву. Матриця A(4,5) матриця C(4,5) |

**Приклад виконання (варіант 30)**

Блок-схема:






Лістинг програми:

```
const M: array[1..4,1..5] of real =
      ((1.2,   0.8,   -1.3,  -0.8,  8),
      (4.5,    2.3,   -5.1,  0,     8.1),
      (2.3,    2.6,   10.5,  3.5,   10),
      (9.5,    -3.8,  -3.3,  3.7,   -7.2));

var i,k: integer;
C: array[1..4,1..5] of real;
max: real;
begin
writeln('початкова матриця');
(* вивидемо матрицю M та обчислимо максимум*)
max:=M[1,1];
for i:=1 to 4 do
    begin
    for k:=1 to 5 do
       begin
       write (M[i,k]:9:1 );
       if  M[i,k]>max then max:= M[i,k];
       end;
       writeln;
    end;
writeln('максимум', max:9:1);
writeln('отриманий результат');
(* обчислимо еплементи матриці C та виведемо її*)
 for i:=1 to 4 do
```

```
    begin
    for k:=1 to 5 do
       begin
       C[i,k]:=M[i,k]/max;
       write (C[i,k]:9:1 );
       end;
       writeln;
    end;
end.
```

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритми виконання.
- Навести програму та кінцевий результат обчислення.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №5

# Тема: Робота з записами

*Мета роботи*: вивчення методів роботи з користувацькими типами даних.

**Теоретичні відомості:**

Записи відносяться до структурованих типів даних. Відмінність запису від масиву полягає в тому, що компоненти запису можуть бути різного типу.

Синтаксис запису:

*TYPE індентифікатор типу = RECORD*

Записи містять певну кількість полів, кожному з яких присвоюється певний тип.

Наприклад:

*Type AVTO=RECORD*
*N: INTEGER;*
*NAME : CHAR;поля запису*
*PRICE : REAL; END;*
*Var A: array[1..5] of AVTO;*

Звернення до компонентів запису відбувається наступним чином: Вказується ідентифікатор змінної типу запис , ставиться крапка та назва потрібного поля. Наприклад: A[i].Price; A[i].Name

Розгалуженні структури даних мають той недолік, що вони зберігаються в ОЗП.

**Індивідуальні завдання**

Створити масив записів із вказаними полями (в масиві повинно міститися не менше 10 елементів). Створіть алгоритм роботи та виконайте завдання згідно варіанту.

Варіанти завдання до практичної роботи №5

| Номер варіанта | Зміст завдання |
|---|---|
| 1 | 2 |
| 1 | Дано масив, елементами якого являються записи з наступними полями: запис «Автомобіль», поля «Номер по порядку», «Назва», «Рік випуску», «Ціна». Вивести на екран вихідну таблицю та всі записи, в яких поле «Ціна» більше чи дорівнює 10000. |
| 2 | Дано масив, елементами якого являються записи з наступними полями: запис «Будинок», поля «Номер квартири», «Поверх», «Прізвище |

| | |
|---|---|
| | власника», «Кількість жителів». Вивести на екран вихідну таблицю та прізвища власників квартир на 5 поверсі. |
| 3 | Дано масив, елементами якого являються записи з наступними полями: запис «Житель», поля «Прізвище», «Ім’я», «По-батькові», «Рік народження», «Стать». Вивести на екран вихідну таблицю та прізвища всіх чоловіків. |
| 4 | Дано масив, елементами якого являються записи з наступними полями: запис «Книга», поля «Автор», «Назва», «Рік випуску», «Кількість сторінок». Вивести на екран вихідну таблицю та назви книг, випущених за останні 5 років. |
| 5 | Дано масив, елементами якого являються записи з наступними полями: запис «Студент», поля «Номер по порядку», «Прізвище», «Рік народження», «Форма навчання». Вивести на екран вихідну таблицю та записи зі студентами бюджетної форми навчання. |
| 6 | Дано масив, елементами якого являються записи з наступними полями: запис «Організація», поля «Назва», «Форма власності», «Прізвище директора», «Кількість працівників». Вивести на екран вихідну таблицю та всі дані про фірми державної форми власності. |
| 7 | Дано масив, елементами якого являються записи з наступними полями: запис «Автомобіль», поля «Марка», «Рік випуску», «Ціна», «Власник». Вивести на екран вихідну таблицю та власників всіх автомобілів марки «Мерседес». |
| 8 | Дано масив, елементами якого являються записи з наступними полями: запис «Житель», поля «Прізвище», «Ім’я», «По-батькові», «Рік народження». Вивести на екран вихідну таблицю та дані всіх чоловіків, які цього року повинні іти до армії. |
| 9 | Дано масив, елементами якого являються записи з наступними полями: запис «Адреса», поля «Прізвище жителя», «Місто», «Вулиця», «Будинок», «Квартира», «Номер телефону». Вивести на екран вихідну таблицю та прізвища всіх жителів будинку, вказаного користувачем під час виконання програми. |
| 10 | Дано масив, елементами якого являються записи з наступними полями: запис «Книга», поля «Автор», «Назва», «Рік випуску», «Кількість сторінок». Вивести на екран вихідну таблицю та авторів книг, які містять в собі більше 500 сторінок. |
| 11 | Дано масив, елементами якого являються записи з наступними полями: запис «Автомобіль», поля «Номер по порядку», «Назва», «Рік випуску», «Ціна». Вивести на екран вихідну таблицю та всі записи з автомобілями, які старше 10 років. |
| 12 | Дано масив, елементами якого являються записи з наступними полями: запис «Будинок», поля «Номер квартири», «Поверх», «Прізвище власника», «Кількість жителів». Вивести на екран вихідну таблицю та прізвища жителів вказаного будинку. |
| 13 | Дано масив, елементами якого являються записи з наступними полями: |

| | |
|---|---|
| | запис «Житель», поля «Прізвище», «Ім’я», «По-батькові», «Рік народження», «Стать». Вивести на екран вихідну таблицю та прізвища всіх жінок. |
| 14 | Дано масив, елементами якого являються записи з наступними полями: запис «Книга», поля «Автор», «Назва», «Рік випуску», «Кількість сторінок». Вивести на екран вихідну таблицю та назви книг, випущених в 1999 році. |
| 15 | Дано масив, елементами якого являються записи з наступними полями: запис «Студент», поля «Номер по порядку», «Прізвище», «Рік народження», «Форма навчання». Вивести на екран вихідну таблицю та записи зі студентами, які народилися в високосному році. |
| 16 | Дано масив, елементами якого являються записи з наступними полями: запис «Організація», поля «Назва», «Форма власності», «Прізвище директора», «Кількість працівників». Вивести на екран вихідну таблицю та всі дані про фірми , в яких працює більше 10 людей. |
| 17 | Дано масив, елементами якого являються записи з наступними полями: запис «Автомобіль», поля «Марка», «Рік випуску», «Ціна», «Власник». Вивести на екран вихідну таблицю та власників всіх автомобілів, які дорожчі 10000 та випущені не пізніше 2010 року. |
| 18 | Дано масив, елементами якого являються записи з наступними полями: запис «Людина», поля «Прізвище», «Ім’я», «По-батькові», «Число», «Місяць», «Рік народження», . Вивести на екран вихідну таблицю та дані всіх людей, що народилися восени. |
| 19 | Дано масив, елементами якого являються записи з наступними полями: запис «Адреса», поля «Прізвище жителя», «Місто», «Вулиця», «Будинок», «Квартира». Вивести на екран вихідну таблицю та прізвища всіх жителів будинку, вказаного користувачем під час виконання програми. |
| 20 | Дано масив, елементами якого являються записи з наступними полями: запис «Книга», поля «Автор», «Назва», «Рік випуску», «Кількість сторінок». Вивести на екран вихідну таблицю та авторів книг, які містять в собі менше 100 сторінок. |
| 21 | Дано масив, елементами якого являються записи з наступними полями: запис «Автомобіль», поля «Номер по порядку», «Назва», «Рік випуску», «Ціна». Вивести на екран вихідну таблицю та всі записи, в яких поле «Ціна» від 20000 до 40000. |
| 22 | Дано масив, елементами якого являються записи з наступними полями: запис «Будинок», поля «Номер квартири», «Поверх», «Прізвище власника», «Кількість жителів». Вивести на екран вихідну таблицю та дані власників квартир на 1 та 2 поверхах. |
| 23 | Дано масив, елементами якого являються записи з наступними полями: запис «Житель», поля «Прізвище», «Ім’я», «По-батькові», «Рік народження», «Стать». Вивести на екран вихідну таблицю та дані всіх чоловіків, які старше 40 років. |

| | |
|---|---|
| 24 | Дано масив, елементами якого являються записи з наступними полями: запис «Книга», поля «Автор», «Назва», «Рік випуску», «Кількість сторінок». Вивести на екран вихідну таблицю та дані книг, випущених в 2000 році. |
| 25 | Дано масив, елементами якого являються записи з наступними полями: запис «Студент», поля «Номер по порядку», «Прізвище», «Курс», «Група». Вивести на екран вихідну таблицю та записи зі студентами 3 курсу. |
| 26 | Дано масив, елементами якого являються записи з наступними полями: запис «Організація», поля «Назва», «Форма власності», «Прізвище директора», «Кількість працівників». Вивести на екран вихідну таблицю та назви фірм державної форми власності. |
| 27 | Дано масив, елементами якого являються записи з наступними полями: запис «Автомобіль», поля «Марка», «Тип кузова», «Рік випуску», «Ціна», «Власник». Вивести на екран вихідну таблицю та власників всіх автомобілів типу «Седан». |
| 28 | Дано масив, елементами якого являються записи з наступними полями: запис «Житель», поля «Прізвище», «Ім'я», «По-батькові», «Рік народження». Вивести на екран вихідну таблицю та дані всіх чоловіків, які молодше 5 років. |
| 29 | Дано масив, елементами якого являються записи з наступними полями: запис «Працівник», поля «Прізвище», «Посада», «Заробітна платня». Вивести на екран вихідну таблицю та прізвища всіх працівників, заробітна платня яких не менше 5000. |
| 30 | Дано масив, елементами якого являються записи з наступними полями: запис «Книга», поля «Автор», «Назва», «Рік випуску», «Кількість сторінок». Вивести на екран вихідну таблицю та авторів книг, які містять в собі більше 500 сторінок. |

**Контрольні запитання:**

1. Дані яких типів не можуть бути елементами запису?
2. В чому відмінність між масивом та записом? Чи можна масим оголосити як запис, у якому всі поля мають однаковий тип?

**Приклад виконання (варіант 30)**

```
type  book= record (*створюємо тип даних «запис»*)
avtor : string[10];
nazva : string[10];
god : integer;
stor : integer;
end;
const n=10;
```

```
var i: integer;
sp_book : array [1..n]  of book;
flag : boolean;
    begin
      flag:=false;
      writeln('введіть список');
      for i:=1 to n do
        begin
        writeln ('введіть  прізвище автора');
        readln (sp_book[i].avtor);
        writeln ('введіть  назву книги ');
        readln (sp_book[i].nazva);
        writeln ('введіть  рік випуску');
        readln (sp_book[i].god);
        writeln ('введіть  кількість сторінок у книзі');
        readln (sp_book[i].stor);
        writeln ('введіть  наступний елемент списку');
        end;
      cls;
      writeln(' вихідний список');
      for i:=1 to n do
        writeln   (sp_book[i].avtor,'   ',sp_book[i].nazva,'      ',sp_book[i].god,   '
      ',sp_book[i].stor );
      writeln(' прізвища авторів, в книзі яких більше 500 сторінок ');
      for i:=1 to n do
        if  sp_book[i].stor>500 then
            begin
            flag:=true;
            writeln (sp_book[i].avtor);
            end;
      if flag=false then writeln ('net');
    end.
```

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритми виконання.
- Навести програму та кінцевий результат обчислення.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №6

# Тема: Робота з типізованими файлами

*Мета роботи*: вивчення методів роботи з файлами, що містять в собі дані заданого типу.

**Теоретичні відомості (для практичних робіт №6 та №7):**

Файл — це самостійна послідовність символів, записана в постійну пам'ять комп'ютера. Це певна виділена область інформації.

Існування файлів не залежить від роботи якої-небудь програми і вони нікуди не зникають навіть при включенні і виключенні комп'ютера.

Файли можуть зберігати в собі різну інформацію.

Описання файлових змінних:

У розділі опису наступний запис

*var*

*f1, f2: text;*

говорить про те, що змінні f1 і f2 це текстові файли (які ми можемо відкрити за допомогою блокнота, подивитися, і при необхідності редагувати).

Встановлення зв’язку між змінною та файлом:

*assign(f,'ім'я_файлу');* — встановлює зв'язок між файловою змінною і самим файлом. Фактично ми говоримо, що змінній f відповідає таке-то ім'я файлу. Тут мається на увазі, що файл з самою програмою знаходиться в тій же теці, що і текстовий файл. Якщо ж він в іншій теці, то нам треба вказати відносне розташування цього файлу.

‘Ім'я_файлу' — повний шлях до вказаного файлу, записаний у відносній або абсолютній формі.

Після того, як ми встановили відповідність файлової змінної і самого файлу ми можемо починати виконувати різні операції над цим файлом:

· Відкрити для читання reset(f); Якщо файлу не існує, то буде виведена помилка.

· Створити або перезаписати rewrite(f); Якщо файлу не існує, то він буде створений;

· Створити або відкрити і дописати в кінець файлу append(f); Якщо файлу не існує, то він буде створений (тільки для текстових файлів);

· Зберегти і закрити файл close(f);

Те, що ми тільки що перерахували - це просто зовнішня робота з файлом.

Як використати або змінити вміст файлу?

Для того що б робити які-небудь внутрішні зміни в самому файлі так само існують команди

· *read(f, a, b);* — читати з файлу f дві змінні a і b. Після виконання

цієї процедури покажчик у файлі пересунеться за змінну b;

· *readln(f, a, b, c);* — читати з файлу f три змінні a, b і c, а потім перекласти покажчик (курсор) на початок наступного рядка; якщо окрім вже лічених змінних в рядку містилося ще щось, то воно буде проігноровано.

· *write(f, a, b, c);* — записати у файл f змінні a, b і c;

· *writeln(f, a, b);* — записати у файл f змінні a і b, а потім записати туди ж символ "кінець рядка".

Зауваження: команди readln та writeln можна застосовувати лише для текстових файлів.

Більш повний перелік команд для роботи з файлами дивись в таблиці.

| **Ім'я та параметри** | **Процедура чи функція** | **Тип параметрів** | **Дії** |
|---|---|---|---|
| `Assign(f,name)` | процедура | `f` - змінна файлового типу, `name` - **`string`** | Пов'язує файлову змінну `f` з файлом з іменем `name` |
| `Reset(f)` | процедура | `f` - змінна файлового типу | Відкриває файл, що пов'язаний з файловою змінною `f` за допомогою процедури `Assign`. Файл при цьому повинен існувати. Текстові файли відкриваються тільки на читання, типізовані – на читання та запис |
| `Rewrite(f)` | процедура | `f` - змінна файлового типу | Створює та відкриває файл, що пов'язаний з файловою змінною `f` за допомогою процедури `Assign`. Якщо файл зі вказаним іменем уже існує – він видаляється і замість нього створюється новий файл. Текстові файли відкриваються тільки на запис, типізовані – на читання та запис |
| `Append(f)` | процедура | `f` - змінна типу `Text` | Відкриває текстовий файл на дозапис. Файловий вказівник встановлюється в кінець файла. |
| `Close(f)` | процедура | `f` - змінна файлового типу | Закриває файл |
| `FileExists(name)` | функція | `name` - **`string`** | Повертає `True`, якщо на диску є файл з іменем name, інакше повертає `False` |
| `CanCreateFile(name)` | функція | `name` - **`string`** | Повертає `True`, якщо можно створити файл с именем name, інакше повертає `False` |
| `Read(f,a,b,...)` | процедура | `f` - змінна файлового типу, `a,b` - змінні простого типа, | Зчитує значення із файлу `f` в змінні `a, b`. Якщо файл типізований, то типи змінних `a, b` ... повинні співпадати з базовим типом файлу, а |

| | | | |
|---|---|---|---|
| | | типу **string** чи вказівники | їх значення зчитуються із файлу в двійковому вигляді. Якщо файл текстовый, то змінні a, b ... можуть мати різні типи, а їх значення повинні зберігатися в файлі в текстовом виді |
| Write(f,a,b,...) | процедура | f - змінна файлового типу, a,b - змінні простого типу, типу **string** чи указатели | Записує значення a, b ... в файл f. Якщо файл типізований, то типи значень a, b ... повинні бути сумісними з базовим типом файлу. Якщо файл текстовый, то значення a, b ... виводяться в нього в текстовом виді, при цьому можна використовувати формати виводу |
| Readln(f,a,b,...) | процедура | f - змінна типу Text, a,b - змінні простого типу, типу **string** чи вказівник | Зчитує значення з текстового файлу f в змінні a, b ..., після чього пропускає символи до кінця строки. Виклик readln(f) просто пропускає символи до кінця строки |
| Writeln(f,a,b,...) | процедура | f - змінна типу Text, a,b - змінні простого типу, типу **string** чи вказівника | Записує значення a, b ... в текстовий файл f, після чього записує в нього символ кінця строки. Значення a, b ... записуються в файл в текстовому виді, при цьому можна використовувати формати виводу. Виклик writeln(f) просто записує в файл символ кінця строки |
| Eof(f) | функція | f - змінна файлового типу | Повертає True, якщо файловий вказівник стоїть на конці файла, і False інакше |
| Eoln(f) | функція | f - змінна типу Text | Повертає True, якщо файловий вказівник стоїть на кінці строки, і False інакше |
| SeekEof(f) | функція | f - змінна типу Text | Пропускає пробіли, символи табуляції і перехода на нову строку, після чього повертає True, якщо файловий вказівник стоїть на кінці файла, і False інакше |
| SeekEoln(f) | функція | f - змінна типу Text | Пропускает пробіли, символи табуляції, після чього повертає True, якщо файловый вказівник стоїть на кінці строки, і False інакше |
| FileSize(f) | функція | f - змінна типу **file** | Повертає кількість елементів в типізованому файлі |
| FilePos(f) | функція | f - змінна типу **file** | Повертає позицію файлового вказівника в типізованому файлі (нумерація елементів в файлі починається з нуля) |

| | | | |
|---|---|---|---|
| `Seek(f,n)` | процедура | `f` - змінна типу **`file`** | Переміщує файловый вказівник в типізованому файлі на n-тий елемент (нумерация починається з нуля) |
| `Truncate(f)` | процедура | `f` - змінна типу **`file`** | Видаляє всі елементи типізованого файла від файлового вказівника до кінця файлу |
| `Rename(f,name)` | процедура | `f` - змінна файлового типу, `name` - **`string`** | Переіменовує файл, що пов'язаний з файловою змінною `f`. Файл повинен бути закритим |
| `Erase(f)` | процедура | `f` - змінна файлового типу | Видаляє файл, що пов'язаний з файловою змінною `f`. Файл повинен бути закритим |

**Індивідуальні завдання:**

У текстовому файлі зберігаються дані у вигляді записів вказаного формату (формат запису визначається по таблиці 1 по останній цифрі номера в журналі).

Таблиця 1.

| 0 - Співробітник | 1- Студент | 2 - Автомобіль | 3 - Книга | 4 – Вузол пристрою |
|---|---|---|---|---|
| Прізвище<br>Посада<br>Відділ<br>**Стаж**<br>Оклад | ВУЗ<br>Група<br>Прізвище<br>Рік народження<br>**Середній бал** | Тип<br>Фірма<br>Обсяг двигуна<br>Пробіг<br>**Ціна** | Автор<br>Назва<br>Видавництво<br>**Рік випуску**<br>Кіл.сторінок | Пристрій<br>Номер вузла<br>Кіл.вузлів<br>**Вага**<br>Ціна |
| 5 - Місто | 6 - Країна | 7 - Підприємство | 8 - Будинок | 9 - Програма |
| Назва<br>Країна<br>**Площа**<br>Кіл.мешканців<br>Щільність нас. | Назва<br>Материк<br>Столиця<br>Кіл.мешканців<br>**Площа** | **Галузь**<br>Назва<br>Рік заснування<br>Голов.продукт<br>Кіл.робітників | Вулиця<br>Номер<br>Кіл.поверхів<br>Кіл.квартир<br>**Кіл.мешканців** | **Мова**<br>Автор<br>Обсяг памяті<br>Час виконання |

Над даними, що знаходяться в файлі, виконати дії згідно варіанту (ключове поле виділено напівжирним підкресленим шрифтом). Номер варіанту визначається по останній цифрі номера студента в списку в журналі.

Таблиця 2.

| № | Завдання |
|---|---|
| 0 | Визначити запис з найбільшим значенням ключового поля. Вивести такі елементи, для яких значення ключового поля дорівнює максимальному. |
| 1 | Визначити середнє значення по ключовому полю. Вивести ті елементи, для яких значення ключового поля менше середнього. |
| 2 | Визначити середнє значення по ключовому полю. Вивести ті елементи, для яких значення ключового поля більше середнього. |
| 3 | Відсортувати дані по спаданню по ключовому полю. Вивести на екран 3 найбільші значення. |
| 4 | Відсортувати дані по зростанню по ключовому полю. Вивести на екран 3 найменші значення. |

| | |
|---|---|
| 5 | Знайти найбільший та найменший елементи по ключовому полю. Вивести на екран всі записи, які знаходяться між цими значеннями |
| 6 | Визначити запис з найменшим значенням ключового поля. Вивести на екран всі записи, що знаходяться між початком файлу та записом з найменшим значенням ключового поля |
| 7 | Вивести на екран всі записи з заданим значенням ключового поля (необхідне значення ввести з клавіатури). Визначити кількість таких елементів |
| 8 | Визначити запис з найбільшим значенням ключового поля. Вивести на екран всі записи, що знаходяться між записом з найбільшим значенням ключового поля та кінцем файлу |
| 9 | Відсортувати всі записи по зростанню по ключовому полю. Вивести відсортовані записи на екран |

**Контрольні запитання:**

1. Опишіть правила роботи з файлом.

2. Які стандартні процедури та функції можна виконати лише над типізованими файлами?

3. Яким чином можна переставити файловий вказівник до вказаної позиції в файлі? Вказати принцип роботи з типізованими та текстовими файлами.

**Приклад виконання:**

У текстовому файлі зберігаються дані у вигляді записів: Студент (ВУЗ, Група, Прізвище, Рік народження, Середній бал успішності). Знайти середнє значення по полю «Середній бал успішності», вивести на екран ті записи, у яких ключове поле більше середнього значення.

Зауваження: студент (по бажанню) може створити файл з записами даного типу заздалегідь (окремим програмним модулем), після чого в іншій програмі його опрацювати, при цьому потрібно привести лістинг обох програм. Крім того, якщо це не вказано явно в завданні, вивід даних може проводитися або на екран, або в файл.

Лістинг програми:

```
type student=record
VuZ:string[20] ;
group :integer;
familiya :string[20];
god_r : integer;
sr_ball : integer;
end;
var r: file of student;
```

```
  i,n:integer; x: student;
  sum,sr:real;
         begin
    n:=4;  sum:=0;
   assign(r,'ept.txt');
   rewrite(r);
  for i:=1 to n do
          begin
   writeln('введите название вуза, номер группы, фамилию, год рождения,
средний балл');
   readln(x.VuZ,x.group,x.familiya, x.god_r,x.sr_ball) ;
  write(r, x);
    end;
  close(r);
   reset (r);    cls;
   while not(eof(r)) do
         begin
   read(r,x) ;
    sum:=sum+x.sr_ball;
         end;
     sr:=sum/n;
     writeln('среднее значение= ',sr);
  seek(r,0);
     writeln('записи, в которых средний балл больше среднего значения  ');
    while not(eof(r)) do
    begin
 read(r,x) ;
  if   x.sr_ball > sr  then writeln(x.VuZ,' ',x.group,' ',x.familiya,' ', x.god_r,'
',x.sr_ball);
end;
 close(r);
end.
```

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритми виконання.
- Навести програму та кінцевий результат обчислення.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №7

# Тема: Робота з текстовими файлами.

*Мета:* навчитися обробляти текстові файли в середовищі програмування Pascal.

**Індивідуальні завдання:**

1) та 16) Дано текстовий файл. Необхідно переписати його вміст в інший текстовий файл, додаючи в початок кожного рядка її порядковий номер (десяткову запис цілого числа шириною в 5 символів) і пробіл.
2) та 17) Дано текстовий файл. Необхідно переписати в інший текстовий файл всі його непусті рядки, які починаються і закінчуються однаковим символом.
3) та 18) Дано текстовий файл. Необхідно переписати в інший текстовий файл його вміст, крім рядків, що починаються з точки.
4) та 19) Дано текстовий файл. Знайти кількість порожніх рядків у ньому.
5) та 20) Дано текстовий файл. Знайти довжину його максимальної рядка.
6) та 21) Дано текстовий файл. Знайти кількість рядків у ньому, що починаються з заданого символу.
7) та 22) Дано текстовий файл. Знайти кількість рядків у ньому, що закінчуються заданих символом.
8) та 23) Дано текстовий файл. Видалити в ньому усі рядки заданої довжини.
3) та 18) Дано текстовий файл. Знайти кількість літер «я» в ньому.
10) та 25) Дано текстовий файл. Переписати його вміст в інший текстовий файл, вставляючи символи переводу рядка після кожної точки.
11) та 26) Дано текстовий файл. Переписати його вміст в інший текстовий файл, доповнюючи праворуч всі його рядки, менші за довжиною 20 символів, пробілами праворуч до 20 символів.
12) та 27) Дано послідовність символів, що закінчується крапкою і вводиться користувачем з клавіатури. Необхідно записати її в текстовий файл, вставляючи в неї символи переводу рядка через кожні 40 символів.
13) та 28) У текстовий файл необхідно записати наступну інформацію:

X
X X
X X X
X X X X
X X X X X
X X X X X X
X X X X X X X
……..
X X X X X X X X ... X (40 елементів)

14) та 29) У текстовому файлі записана послідовність цілих чисел, розділених пробілами. Переписати в інший текстовий файл всі числа з першого файлу, крім чисел, рівних 29.
15) та 30) Дано текстовий файл, що складається з одного або декількох рядків. Кожен рядок файлу містить числа, розділені пробілами. Перевірити, чи міститься у файлі задане число. Результати з відповідними коментарями дописати в вихідний файл.

**Контрольні запитання:**

1. Якими стандартними функціями можна користуватися лише при обробці текстових файлів:?

2. Вкажіть символ закінчення строки в текстовому файлі?

**Приклад виконання:**

Дано текстовий файл. Переписати з одного текстового файлу в інший строчки, при чому, якщо довжина строчки менше 20 символів, дописати до 20 символів пробілами на початку строчки.

```
var f,f1:text;
s:string; i,k: integer;
begin
   assign(f,'file1.txt');
   assign(f1,'file2.txt');
   reset (f);
   rewrite(f1);
   while not(eof(f))   do begin
   readln(f,s);
   if length(s)<20 then
                begin
                for i:=1 to 20 - length(s) do
                write (f1,' ');  end;
   writeln(f1,s)  ; end;
 close(f1);
 close(f);
end.
```

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритми виконання.
- Навести програму та кінцевий результат обчислення.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №8

# Тема: Робота з процедурами та функціями.

*Мета:* навчитися створювати процедури (функції) в середовищі програмування Pascal.

**Теоретичні відомості:**

Підпрограмою називається іменована логічно завершена група операторів алгоритмічної мови, яку можна викликати для виконання по її імені будь-яку кількість раз з різних місць програми.

В мові програмування Паскаль для організації підпрограм використовуються процедури і функції.

Процедура - це незалежна іменована частина програми, призначена для виконання певних дій. Складається із заголовку і тіла.

Після однократного опису процедури її можна викликати по імені з будь-якої частини програми. Коли процедура закінчить своє виконання, програма продовжиться з оператора, який йде безпосередньо за оператором виклику процедури. Використання імені процедури в програмі називається оператором процедури або викликом процедури. Ім'я процедури не може знаходитись у виразі в якості операнда.

Функція аналогічна процедурі, але є відмінність: ім’я функції входить у вираз як операнд і функція повертає одне значення, тоді як процедура може повертати декілька значень або зовсім їх не повертати.

Всі процедури і функції діляться на стандартні і визначені користувачем. Стандартні процедури і функції є частиною мови і можуть викликатися по імені без їх попереднього опису в розділі процедур і функцій.

Процедури користувача являють собою іменовану групу операторів, які реалізують певну частину загальної задачі і викликаються при необхідності для виконання з будь-якої позиції розділу операторів. Структура процедури майже співпадає із структурою програми

- заголовок процедури;
- розділ модулів;
- розділ опису міток;
- розділ опису констант;
- розділ опису типів;
- розділ опису змінних;
- розділ опису процедур і функції;
- розділ операторів.

Заголовок процедури складається із зарезервованого слова Procedure, ідентифікатора (імені) процедури і необов'язкового розміщеного в круглих дужках списку формальних параметрів з вказівкою їх типів. Ім'я процедури

повинно бути унікальним в межах всієї програми.

Формат:

*Procedure <ім'я>[(<формальні параметри>)];*

Приклад.

Procedure Screen;{без параметрів}

Procedure Sum(a,b:integer);{a і b - формальні параметри}

Всі інші розділи процедури повністю аналогічні розділам програми, тільки розділ операторів закінчується крапкою з комою, а не крапкою (як у програми).

Для звертання до процедури використовується оператор виклику процедури. Він складається з ідентифікатора (імені) процедури і списку фактичних параметрів, розділених комами і розміщених у круглих дужках. Список параметрів може бути відсутнім , якщо процедурі не передається ніяких значень.

Формат:

*<ідентифікатор процедури>[(<фактичні параметри>)];*

Приклад.

Screen; { без параметрів }

Sum (a,b+a); {з двома параметрами a і b+a }

При виконанні процедури формальні параметри замінюються на фактичні. Це дозволяє виконувати процедуру з різними початковими даними. Кількість і тип фактичних параметрів повинні співпадати з кількістю і типом формальних параметрів. Якщо процедура вертає у програму якість значення, то відповідні формальні параметри повинні бути описані як параметри - змінні з використанням службового слова Var.

Функція, визначена користувачем має ту ж саму структуру, що й процедура.

Заголовок функції містить зарезервоване слово Function, ідентифікатор (ім'я) функції, необов'язковий розміщений у круглих дужках список формальних параметрів з вказівкою їх типів і типу означення, яке повертається функцією. Ім'я функції повинно бути унікальним в межах всієї програми.

Формат.

*Function <ім'я>[(<формальні параметри>)]:<тип результату>*

Приклад.

Function Logic:boolen; {без параметрів}

Function Max(x,y:real) {з параметрами x і y}

В розділі операторів функції повинен бути хоча б один оператор присвоєння ідентифікатору функції значення. Якщо таких присвоєнь декілька, то результатом виконання функції буде значення останнього

оператора присвоєння.

Звертання до функцій здійснюється по імені з вказівкою її фактичних параметрів (якщо вони є). Тип і кількість фактичних параметрів повинні відповідати типу і кількості формальних параметрів, вказаних у заголовку.

*Формат. <ідентифікатор функції> [(<фактичні параметри>)]*

**Індивідуальні завдання:**

1) Дано відрізки a, b, c і d. Для кожної трійки цих відрізків, з яких можна побудувати трикутник, обчислити площу даного трикутника. (Визначити процедуру print_square (x, y, z), що друкує площу трикутника зі сторонами x, y, z, якщо такий трикутник існує.)

2) Дано довжини a, b і c сторін деякого трикутника. Знайти медіани трикутника, сторонами якого є медіани вихідного трикутника. (Зауваження: довжина медіани, проведеної до сторони a, дорівнює 0.5 * sqrt (2 * b ^ 2 + 2 * c ^ 2 - a ^ 2))

3) Дано 30-елементні дійсні вектори x, y і z. Обчислити величину (z, z) - (b, c), де z позначає той з цих векторів, в якому найбільший мінімальний елемент (вважати, що такий вектор єдиний), b і c позначають два інших вектора, a (p, q ) - скалярний добуток p і q.

4) Дано дві квадратні дійсні матриці 10-го порядку. Надрукувати ту з них, в якій найменший слід (сума діагональних елементів), вважаючи, що така матриця одна.

5) Дана непорожня послідовність слів, у кожному з яких від 1 до 6 латинських букв; між сусідніми словами - кома, за останнім словом - крапка. Надрукувати ті слова, у яких однакові "сусіди", тобто збігаються попереднє і наступне слова. (Визначити процедуру readword (w), яка вводить чергове слово і привласнює його 6-литерному рядку w, а кому або крапку присвоює деякій глобальній змінній.)

6) Дано три дійсні квадратні матриці 4-го порядку. Надрукувати ту з них, норма якої найменша (вважати, що така матриця одна). В якості норми матриці взяти максимум абсолютних величин її елементів.

7) Дано координати вершин двох трикутників. Визначити, який з них має найбільшу площу.

8)
```
const
   n = 8;
   m = 13;
  type matrix = array [1 .. n, 1 .. m] of real;
```

Описати процедуру swap (A, B), яка міняє місцями максимальні елементи матриць A і B. (Вважати, що в кожній матриці тільки один максимальний елемент.)

9) Дано коефіцієнти многочленів P (x) і Q (x) 5-го ступеня та дано дійсне число a. Обчислити величину P (a + Q (a) P (a +1)).

10) Дано три цілі матриці розміром 9x4. Надрукувати ту з них, де більше нульових рядків (якщо таких матриць кілька, надрукувати їх усі).

**Приклад виконання :**
Скласти програму для обчислення факторіалу числа F=n! з використанням функції для обчислення факторіалу (n!=1*2*3*...*n).

```
Program PR8_1;
Var F,n:Integer;
Function Factorial(n:integer):longint;
Begin
 if n=1 then Factorial:=1
 else Factorial:=n*Factorial(n-1)
End;
Begin
Write ('Введіть N');
ReadLn(n);
F:=Factorial(n);
WriteLn ('F(',n,')=',F);
End.
```

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритми виконання.
- Навести програму та кінцевий результат обчислення.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №9

# Тема: Робота з графічними примітивами.

*Мета:* навчитися працювати з графікою в середовищі програмування Pascal.

**Теоретичні відомості:**
Модуль GraphABC містить константи, типи, процедури, функції та класи для виконання зображень в графічному вікні. Вони поділені на наступні категорії:
Графічні примітиви ( служать для виводу на екран стандартної фігури:

```
procedure SetPixel(x,y,color: integer);
function GetPixel(x,y): integer;
procedure MoveTo(x,y: integer);
procedure LineTo(x,y: integer);
procedure Line(x1,y1,x2,y2: integer);
procedure Circle(x,y,r: integer);
procedure Ellipse(x1,y1,x2,y2: integer);
procedure Rectangle(x1,y1,x2,y2: integer);
procedure RoundRect(x1,y1,x2,y2,w,h: integer);
procedure Arc(x,y,r,a1,a2: integer);
procedure Chord(x,y,r,a1,a2: integer);
procedure TextOut(x,y: integer; s: string);
procedure FloodFill(x,y,color: integer);
procedure FillRect(x1,y1,x2,y2: integer);)
```

Дії з кольором ( установка та зчитування параметрів кольору)
Дії з пером (олівцем) (служать для установки параметрів та стилів ліній:

```
function PenX: integer;
function PenY: integer;
procedure SetPenColor(color: integer);
function PenColor: integer;
procedure SetPenWidth(w: integer);
function PenWidth: integer;
procedure SetPenStyle(ps: integer);
function PenStyle: integer;)
```

Дії з пензлем (служать для установки параметрів та стилів заливки:

```
procedure SetBrushColor(color: integer);
function BrushColor: integer;
procedure SetBrushPicture(fname: string);
procedure ClearBrushPicture;
procedure SetBrushStyle(bs: integer);
function BrushStyle: integer;)
```

Дії з шрифтом ( установка параметрів шрифту:

```
procedure SetFontColor(color: integer);
```

```
  function FontColor: integer;
  procedure SetFontSize(sz: integer);
  function FontSize: integer;
  procedure SetFontName(name: string); (по замовчуванню
  - MS Sans Serif)
  function FontName: string;
  procedure SetFontStyle(fs: integer);
  function FontStyle: integer;
```

**Стилі шрифта:**

fsNormal – звичайний;
fsBold – напівжирний;
fsItalic – курсив;
fsBoldItalic – напівжирний курсив;
fsUnderline – підкреслений;
fsBoldUnderline – напівжирний підкреслений;
fsItalicUnderline – курсив підкреслений;
fsBoldItalicUnderline – напівжирний курсив підкреслений.

```
  function TextWidth(s: string): integer;
  function TextHeight(s: string): integer;
```

Дії з малюнками: клас Picture ( робота з файлами зображень:

```
  function LoadPicture(fname: string): integer;
  n:=LoadPicture(fname);
  procedure SavePicture(n: integer; fname: string);
  procedure DrawPicture(n,x,y: integer);
  procedure DrawPicture(n,x,y,w,h: integer);
  procedure DrawPicture(n: integer; x,y: integer; r:
  Rect);
  procedure DrawPicture(n: integer; x,y,w,h: integer;
  r: Rect);
  procedure CopyRect(n: integer; dest: Rect; n1:
  integer; src: Rect);
  procedure DestroyPicture(n: integer);
  procedure SetPictureSize(n,w,h: integer);
  function PictureWidth(n: integer): integer;
  function PictureHeight(n: integer): integer;
  function PictureTransparent(n: integer): boolean;
  function CreatePicture(w,h: integer): integer;
  procedure FlipPictureHorizontal(n);
  procedure FlipPictureVertical(n);
```

Дії з графічним вікном ( обробка вікна виводу:

```
  procedure ClearWindow(c: ColorType);
  function WindowWidth: integer;
  function WindowHeight: integer;
  function WindowLeft: integer;
```

```
function WindowTop: integer;
function WindowCaption: string;
procedure SetWindowWidth(w: integer);
procedure SetWindowHeight(h: integer);
procedure SetWindowLeft(l: integer);
procedure SetWindowTop(t: integer);
procedure SetWindowSize(w,h: integer);
procedure SetWindowPos(l,t: integer);
procedure SetWindowCaption(s: string);
procedure SetWindowTitle(s: string);
procedure SaveWindow(fname: string);
procedure LoadWindow(fname: string);
procedure FillWindow(fname: string);
procedure FillWindow(n: integer);
procedure CloseWindow;
function ScreenWidth: integer;
function ScreenHeight: integer;
procedure CenterWindow;
procedure MaximizeWindow;
procedure NormalizeWindow;
```

Процедури та функції аналогічні класу TCanvas в Delphi.

**Індивідуальні завдання:**

На 3 бали: використовуючи модуль crt у вказаному місці екрану вивести свої прізвище, ім'я та по-батькові з інтервалом 2 секунди.

На 4 бали: використовуючи модуль GraphABC виконати на екрані малюнок, використовуючи не менше 5 стандартних фігур та 5 кольорів.

На 5 балів: додати до малюнку ефекти руху.

**Контрольні запитання:**

1. Яким чином можна вивести на екран малюнок з зовнішнього файлу? Які формати малюнків підтримує середовище програмування?
2. Яким чином можна вивести у вказану область вікна текст з заданими параметрами шрифту, розміром тощо.
3. Яким чином можна запрограмувати зміну розміщення зображення на екрані?

**Приклад виконання:**

Вивести на екран в випадковому місці кола, що поступово збільшуються.

```
uses GraphABC;

const
  speed=2;
```

```
procedure Kaplia(x0,y0: integer);
var
  i,r: integer;
begin
  r:=1;
  for i:=0 to 63 do
  begin
    SetPenColor(RGB(i*4,i*4,i*4));
    Circle(x0,y0,r);
    if i mod speed = 0 then Sleep(10);
    r:=r+2;
    SetPenColor(clWhite);
    Circle(x0,y0,r);
  end;
end;

const z=50;

begin
  while True do
    Kaplia(Random(WindowWidth-2*z)+z,Random(WindowHeight-2*z)+z);
end.
```

**Оформлення звіту:**

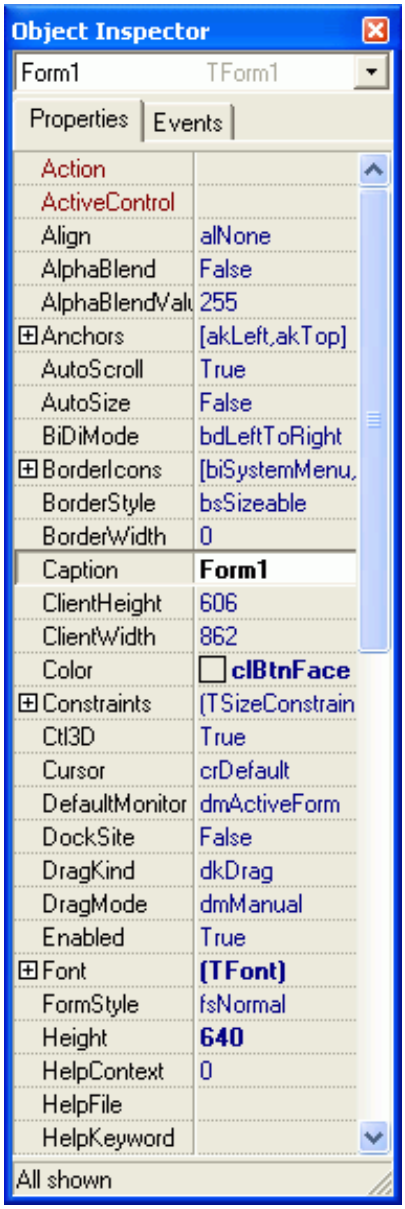
- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести програму та зробити знімок екрану під час виконання.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №10
# Тема: Робота з компонентами сторінки Standard.

*Мета роботи*: Навчитися працювати з формою, організовувати введення та виведення даних.

**Теоретичні відомості.**

Розглянемо основні властивості компонентів. Почнемо ми з базового елементу будь-якого віконного додатки - форми. Властивостей у форми досить багато, і в різних версіях Delphi їх набір може трохи відрізнятися. Тут буде розгляд властивостей на основі Delphi 7 .




## Властивості форми (TForm) у Object Inspector

**Action** - визначає об'єкт TAction. Це об'єкт служить для швидкої прив'язки дій до компонентів, особливо - до пунктів меню і панелям інструментів. Але може бути прив'язаний і до форми. Для управління TAction служать редактори TActionList зі сторінки Standard і TActionManager зі сторінки Additional.

**ActiveControl** - визначає елемент, який має в даний момент фокус введення. Якщо вибрати який-небудь об'єкт під час розробки (design - time) , то при запуску програми цей об'єкт і буде мати фокус вводу. Також властивість може бути корисною і під час виконання ( run - time ) - можна дізнатися , який об'єкт "тримає " фокус в даний момент, а також можна перемістити фокус на будь-який з об'єктів.

**Align** - визначає вирівнювання форми на екрані. Властивість приймає одне з таких значень:

*alBottom* - по нижньому краю;

*alClient* - вся призначена для користувача (клієнтська) область;

*alCustom* - вирівнювання визначається викликом методом об'єкта-батька;

*alLeft* - по лівому краю;

*alNone* - без вирівнювання;

*alRight* - по правому краю;

*alTop* - по верхньому краю.

**AlphaBlend** - включає / вимикає прозорість форми.

**AlphaBlendValue** - задає ступінь непрозорості форми: 0 - форма повністю невидима, 255 - повністю видима. Прозорість активується тільки

при установці властивості AlphaBlend в True.

Властивості прозорості форми (AlphaBlend, AlphaBlendValue, TransparentColor і TransparentColorValue ) коректно працюють тільки на ОС Windows XP і наступних версіях.

**Anchors** - визначає напрями, за якими компоненти "прив'язуються" до форми.

Приклад: якщо встановити у форми значення akLeft і akRight цієї властивості в True, і точно також зробити у кнопки, то при зміні ширини форми розмір кнопки (ширина) також буде змінюватися.

**AutoScroll** - включає автоматичне поява смуг прокрутки ( Scroll bars ) на формі, коли розмірів форми недостатньо для відображення всіх елементів.

**AutoSize** - включає автоматичну зміна розмірів форми згідно з позиціями розміщених на ній елементів.

**BiDiMode** - визначає двонаправлене відображення елемента. У деяких мовах написання здійснюється не зліва - направо, а навпаки. Це властивість створено саме для цієї мети.

**BorderIcons** - визначає кількість кнопок, які відображаються в заголовку вікна:

*biSystemMenu* - єдиний елемент, який не є кнопкою - відповідає за системне меню вікна, яке викликається комбінацією клавіш [ Alt] + [ Пробіл].

*biMinimize* - кнопка згортання ( мінімізації ) вікна;

*biMaximize* - кнопка розгортання вікна;

*biHelp* - кнопка довідки.

Якщо хоча б одна з кнопок згортання і розгортання включена, то незалежно від стану іншої, відображаються обидві (але друга неактивна). Якщо вимкнені обидві, вони не відображаються взагалі. Це не залежить від Delphi - так влаштована ОС Windows.

**BorderStyle** - визначає поведінку кордонів вікна і загальний тип вікна:

*bsDialog* - діалогове вікно ( з кнопок - тільки "Закрити", іконки в заголовку вікна немає) ;

*bsNone* - "чистий аркуш" (відсутність у вікна кордонів і заголовка) - застосовується зазвичай для створення заставок під час запуску програми ;

*bsSingle* - звичайне вікно, але із забороною зміни розмірів;

*bsSizeable* - звичайне вікно (за замовчуванням) - розміри форми можна змінювати;

*bsSizeToolWin* - спрощене вікно із зменшеним заголовком;

*bsToolWindow* - спрощене вікно із зменшеним заголовком без можливості зміни розмірів.

**BorderWidth** - ширина кордону вікна в пікселах. Кордон є невидимою частиною форми.

**Caption** - текст заголовка форми.

**ClientHeight**, **ClientWidth** - розмір клієнтської (користувальницької) частини форми.

**Color** - колір форми.

**Constraints** - визначає мінімальні та максимальні розміри висоти і

ширини форми у точках. 0 - будь-яке значення, тобто без обмежень.

**Ctl3D** - властивість визначає 3D-вид форми. При вимкненому - "плоске" зображення.

**Cursor** - курсор миші в той момент, коли він знаходиться над формою.

**DefaultMonitor** - визначає, на якому моніторі з'явиться форма . Має сенс застосовувати цю властивість тільки за наявності більше, ніж одного монітора (наприклад , якщо кілька екранів) .

**DockSite** , **DragKing** і **DragMode** - визначають поведінку форми при здійсненні операцій Drag & Drop .

**Enabled** - відповідає за загальну активність форми. Якщо встановлено в False, форма недоступна.

**Font** - шрифт, використовуваний на формі.

**FormStyle** - стиль форми або її поведінку в MDI - додатку (багатовіконний додаток, де додаткові форми розташовуються "всередині" основної форми).

**Height** - висота форми у точках. На відміну від ClientWidth є висотою з урахуванням заголовка і кордонів форми.

**HelpContext**, **HelpFile**, **HelpKeyword**, **HelpType** - властивості для зв'язку форми з файлом довідки у форматі *. Hlp.

**HorzScrollBar** - визначає зовнішній вигляд і поведінку горизонтальної смуги прокрутки вікна.

**Icon** - значок (іконка) форми. Відображається в заголовку зліва від заголовка. Задається файлом у форматі *. Ico .

**KeyPreview** - якщо властивість встановлено в True , то при натисканні клавіш спочатку будуть викликатися обробники форми, а тільки потім обробники того компонента, який в даний момент має фокус вводу. Події, пов'язані з натисканням клавіш - OnKeyDown (), OnKeyPress (), OnKeyUp ().

**Left** - позиція форми на екрані (лівого верхнього кута) у точках.

**Menu** - дозволяє вибрати один з компонентів - меню, який стане головним меню вікна , тобто буде відображатися вгорі.

**Name** - ім'я форми як об'єкта. Може містити тільки латинські букви, цифри і знак підкреслення , і не може починатися з цифри. Фактично , це те ім'я, по якому в програмі можна звернутися до форми.

**ObjectMenuItem** - використовується при роботі з OLE - об'єктами і дозволяє зв'язати пункт меню і OLE - об'єкт: коли об'єкт виділено, пункт меню активний і навпаки.

**OldCreateOrder** - визначає, коли відбуваються події OnCreate() і OnDestroy() форми. Якщо встановлено в False, то OnCreate () відбудеться після виклику всіх конструкторів, а OnDestroy () - після виклику всіх деструкторів. Початкове значення - False, змінювати не рекомендується.

**ParentBiDiMode** - зміна властивості BiDiMode згідно значенням об'єкта-предка форми.

**ParentFont** - зміна шрифту ( Font) згідно значенням об'єкта - предка.

**PixelsPerInch** - пропорції шрифту в системі ( точок на дюйм).

**PopupMenu** - дозволяє вказати контекстне меню (об'єкт TPopupMenu)

для форми. Це меню викликається натисненням правої кнопки миші.

**Position** - визначає початкову позицію форми на екрані, тобто в момент її появи.

Основні значення:

*poDesigned* - поява в тому місці, в якому форма розташована в design - time;

*poDesktopCenter* - по центру робочого столу (рекомендоване значення);

*poScreenCenter* - по центру екрану;

*poMainFormCenter* - по центру головної форми додатка (для головної форми не має сенсу).

*PrintScale* - визначає розміри форми при виведенні її зображення на друк.

*Scaled* - включає масштабування форми відповідно із заданим значенням властивості PixelsPerInch.

*ScreenSnap* - якщо встановлено в True , то форма буде автоматично "прилипати" до країв екрану в момент переміщення.

*SnapBuffer* - визначає відстань (у пікселах ) , на якому форма буде "прилипати" до краю екрану .

**ShowHint** - включає / вимикає показ підказки (Hint) .

**Tag** - спеціальна властивість, яке є у всіх об'єктів. Спеціального застосування для цієї властивості немає, тому вона використовується для різних цілей у конкретній ситуації. Властивість зручна в тому випадку , якщо потрібно зберігати деяке ціле число - не доведеться заводити додаткову змінну.

**Top** - позиція форми ( лівого верхнього кута) на екрані в пікселях.

**TransparentColor** - включає / вимикає прозорість певного кольору форми.

**TransparentColorValue** - задає колір , який буде прозорим.

**UseDockManager** - використовується при реалізації Drag & Drop технології, надаючи додаткові можливості цього методу взаємодії .

**VertScrollBar** - визначає зовнішній вигляд і поведінка вертикальної смуги прокрутки вікна.

**Visible** - визначає видимість форми на екрані.

**Width** - ширина вікна в пікселах, включаючи межі.

**WindowMenu** - властивість - аналог властивості Menu, але використовується при створенні MDI -форм.

**WindowState** - один зі станів вікна:

*wsNormal* - звичайний стан (займає частину екрану);

*wsMinimized* - вікно згорнуто;

*wsMaximized* - вікно розгорнуте на весь екран.

У результаті ми отримуємо величезну кількість властивостей, здатних змінити як зовнішній вигляд форми, так і її поведінку, а також поведінку компонентів, розташованих на ній.

Властивості інших візуальних компонентів дуже схожі і велика їх частина просто - напросто збігається.

**Примітки:**

Властивості, назви яких починаються зі слова Parent (англ. - батько), в більшості випадків пов'язують значення деяких властивостей зі значеннями відповідних властивостей об'єкта-батька. Так, кнопка (TButton) має властивість ParentFont і властивість Font, що відповідає за шрифту тексту на цій кнопці. Але й сама форма має властивість Font. У результаті, якщо у кнопки встановити ParentFont в True, а потім змінити шрифт у форми, то шрифт у кнопки зміниться відповідним чином. Це дозволяє швидко змінювати одні й ті ж властивості у великого числа компонентів. Інші подібні властивості - ParentShowHint, ParentColor, ParentBiDiMode.

Властивість Cursor, що відповідає за курсор, є у більшості компонентів. Але при переміщенні курсору його вигляд змінюється на той, який заданий у самого "дальнього" об'єкта. Тобто якщо і форми і у кнопки задані різні форми курсору, то при переміщенні над кнопкою буде використовуватися курсор, заданий у самої кнопки. Число "вкладень" одних компонентів в інші може бути досить великим.

## Вікна повідомлень

### Просте вікно з повідомленням - ShowMessage ( )

Найпримітивніше вікно містить вказаний текст і кнопку OK для закриття вікна. Викликати таке вікно можна процедурою ShowMessage (), параметром якої є текст - він і буде відображений у вікні:

*ShowMessage ( 'текст виводу');*

### Діалогове вікно - MessageDlg ( )

Діалогові вікна - складніший тип вікон. Діалогові вікна часто використовуються для "спілкування" з користувачем.

Створюються діалогові вікна функцією MessageDlg ( ). Це саме функція, а не процедура. Повертається значенням є кнопка, яку натиснув користувач. У функції 4 вхідних параметра :

• Текст повідомлення (тип даних - String ) ;

• Тип діалогового вікна (спеціальний тип даних - TMsgDlgType) - вказує на значок , розташований у вікні і на заголовок вікна . Цей параметр задається одній з наступних констант :

- mtInformation - інформаційне вікно (значок у вікні - буква " і ");
- mtConfirmation - вікно з питанням (значок – знак питання);
- mtWarning - попереджувала вікно (значок - знак оклику);
- mtError - вікно з повідомленням про помилку (значок - хрест на червоному тлі);
- mtCustom - вікно без значка, а в заголовку - назва виконуваного файлу програми (всі інші типи в заголовок поміщають відповідно типу діалогу назва - Information, Warning і т.д.)

• Кнопки , які будуть показані у вікні. Кожній кнопці також відповідає певна константа. Кнопки перераховуються через кому, а весь цей "комплект" обрамляється квадратними дужками. Ось константи всіх доступних кнопок:

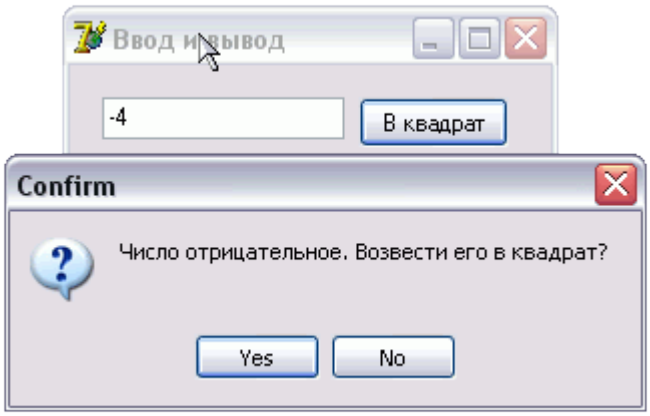
mbYes, mbNo, mbOK, mbCancel, mbAbort, mbRetry, mbIgnore, mbAll, mbNoToAll, mbYesToAll , mbHelp . Назва константи говорить про назву самої кнопки. При бажанні, користувач має право не вказувати ні одну кнопку в вікні.

• Четвертий параметр вказує на індекс розділу в довідковій системі, відповідний даному діалогу. Як правило, не використовується і задається просто 0 .

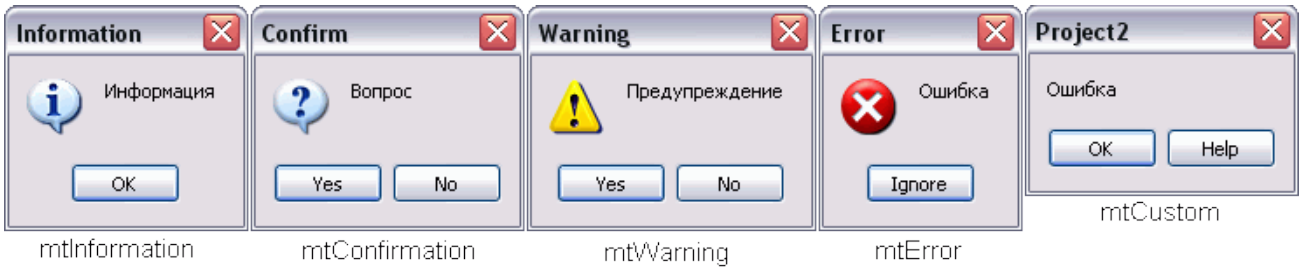
В якості повертаються функцією значень служать все ті ж константи кнопок, тільки з тією відмінністю , що замість " mb " вони починаються на "mr" ( скорочення від "modal button" і "modal result" відповідно).

Приклад:

*MessageDlg('Число отрицательное. Возвести его в квадрат?',mtConfirmation,[mbYes,mbNo],0)*




Зовнішній вигляд вікон усіх типів:

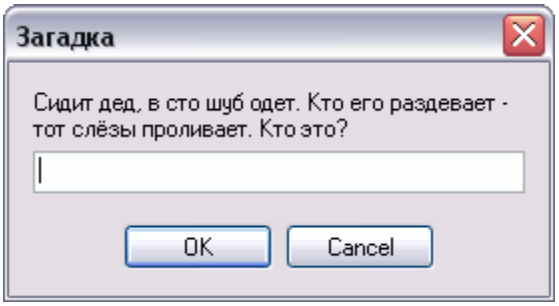



## Введення за допомогою діалогового вікна

Спеціальні віконця існують не тільки для виведення на екран, але і для введення. Прикладом такого вікна є InputBox (). Функції передається 3 параметри: текст заголовка вікна, текст пояснення і значення, що знаходиться в полі при показі вікна на екран. Введений в поле виводу текст функція повертає як значення - результат.

Приклад :

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  if InputBox('Загадка','Сидит дед, в сто шуб одет. Кто его раздевает - тот слёзы проливает. Кто это?','') ='лук' then
    MessageDlg('Правильно!',mtInformation,[mbOk],0)
  else
    MessageDlg('Вы не угадали.',mtWarning,[mbOk],0)
end;
```




**Індивідуальні завдання.**

Створити додаток у середовищі програмування Delphi для обчислення значення функції У (згідно варіанту) в точці, заданій користувачем. Введення даних повинно здійснюватися за допомогою об'єкта «Edit», виведення – на вибір розробника або через Edit, або через Label, або через ShowMessage. Дані, що виводяться, округлити до тисячних. Візуальне оформлення додатку – на вибір студента.

| | |
|---|---|
| 1 | $y = \begin{cases} x^{e^x} + e^2, \text{якщо } 1 \le x < 2 \\ \sin(x+5)^2 + \dfrac{x+0{,}3}{9}, \text{якщо } -1 < x < 1 \\ \sqrt[3]{1+\sqrt{\lvert x-5\rvert}}, \text{якщо } -5 \le x < -1 \end{cases}$ |
| 2 | $y = \begin{cases} ln^2 x^3 + \sqrt{x}, \text{якщо } 3 < x < 6 \\ \dfrac{x}{3+\dfrac{x^2}{5+x}}, \text{якщо } x > 10 \\ x\ ln\ x + arctgx, \text{якщо } 6 \le x \le 10 \end{cases}$ |
| 3 | $y = \begin{cases} 1 + sin^2(x+0{,}1), \text{якщо } -1 < x < 1 \\ \dfrac{x}{12} + \dfrac{x}{\dfrac{x^2}{5+x} + \sqrt{\lvert x\rvert}}, \text{якщо } x \le -1 \\ tgx - \text{в інших випадках} \end{cases}$ |
| 4 | $y = \begin{cases} cos^2\lvert x\rvert, \text{якщо } -2 < x < -1 \\ \sqrt[3]{x + sin^2(x+3)}, \text{якщо } x \le -1 \\ arctg^2\dfrac{\lvert x\rvert}{2}, \text{якщо } x = -5 \end{cases}$ |
| 5 | $y = \begin{cases} \sqrt[3]{x} + e^{x+9}, \text{якщо } 7 < x < 9 \\ tg\sqrt{x}, \text{якщо } x = 3 \\ \ln\lvert x-9\rvert, \text{якщо } x = 1 \text{ або } x = -1 \end{cases}$ |

| | |
|---|---|
| 6 | $y=\begin{cases}\dfrac{5+\sqrt{x}}{\sqrt{100-x}}\text{, якщо } 1<x<2 \text{ або } 4<x<7\\ \lvert x\rvert\text{, якщо } x<-1\\ ln^2x^3\text{, якщо } x=0{,}75\end{cases}$ |
| 7 | $y=\begin{cases}\sqrt[4]{x}+e^{x^2}\text{, якщо } 1<x<1{,}5\\ \sqrt{\ln x}\text{, якщо } 0<x\le 1\\ \dfrac{5+\sqrt{\lvert x\rvert}}{2+2x}\text{, в інших випадках}\end{cases}$ |
| 8 | $y=\begin{cases}\dfrac{1+x^2}{\sqrt{1+x^4}}\text{, якщо } x\le 0\\ 2x+\dfrac{sin^2x}{2+x}\text{, якщо } x>1\\ 3\text{sinx}-cos^2\text{x}\text{, якщо } 0<x\le 1\end{cases}$ |
| 9 | $y=\begin{cases}\frac{x+\frac{x}{2}}{2(x+cos^2x)}\text{, якщо } x\le 0\\ x^{\frac{1}{sinx}}\text{, якщо } x>1\\ 3-\sqrt{1+x^2}\text{, якщо } 0<x\le 1\end{cases}$ |
| 10 | $y=\begin{cases}\sqrt{\lvert x+(x+1)^3\rvert}\text{, якщо } x\le 2\\ \frac{3x^2+\sqrt{\lvert sinx\rvert}}{1+x^2}\text{, якщо } 2<x\le 0\\ 2\sqrt{\lvert 1+2x\rvert}\text{, в інших випадках}\end{cases}$ |
| 11 | $y=\begin{cases}\frac{3+sinx}{1+x^2}\text{, якщо } x\le 0\\ 2x^2cos^2x\text{, якщо } x>1\\ tg^2\frac{x}{2}+\cos(x+2)\text{, якщо } x=0{,}65\end{cases}$ |
| 12 | $y=\begin{cases}\sqrt{1+2x^3-sin^2x}\text{, якщо } x\le 0\\ \dfrac{2+x}{\sqrt[3]{2+e^{-0{,}1x}}}\text{, якщо } x>1\\ 2\cos(x+5)\text{, якщо } 0<x\le 1\end{cases}$ |
| 13 | $y=\begin{cases}\dfrac{x^3}{3+\frac{x^3}{5}}\text{, якщо } x\le -1\\ \sqrt{1+x^2}\text{, якщо } -1\le x<0\\ \dfrac{1+x}{1+\sqrt[3]{1+e^{-0{,}2x}}}\text{, якщо } x=5 \text{ або } x=10\end{cases}$ |
| 14 | $y=\begin{cases}\sqrt[3]{1+\sqrt{\lvert x-10\rvert}}\text{, якщо } x\le -2\\ tg^2x+1\text{ , якщо } x>1\\ 1+(x+\cos(x-1))-\text{в інших випадках}\end{cases}$ |
| 15 | $y=\begin{cases}\sqrt{1+\lvert x\rvert}\text{, якщо } x\le 0\\ \dfrac{1+3x}{2+\sqrt[3]{1+x}}\text{ , якщо } x>1\\ arctg\dfrac{3x}{2}\text{, якщо } x=0{,}33\end{cases}$ |

| | |
|---|---|
| 16 | $y=\begin{cases}\dfrac{\sqrt{1+\lvert x\rvert}}{2+\lvert x\rvert}, \text{якщо } x\le 0\\ \dfrac{1+x}{2+cos^3x}\ , \text{якщо } x>1\\ 4\sqrt{1+x^3}, \text{в інших випадках}\end{cases}$ |
| 17 | $y=\begin{cases}\sqrt[3]{1+x^2}\ , \text{якщо } x\le 0\\ sin^2x+\dfrac{1+x}{1+cos^2x}\ , \text{якщо } x>5\\ tg^2x+ctgx\ , \text{якщо } x=2 \text{ або } x=3\end{cases}$ |
| 18 | $y=\begin{cases}2^{-x}\sqrt{\lvert x\rvert}\ , \text{якщо } x<0\\ \sqrt{e^{x+1}+sinx}\ , \text{якщо } x>1\\ \ln x^2 - \text{в інших випадках}\end{cases}$ |
| 19 | $y=\begin{cases}cos^3(x+3)\ , \text{якщо } x<0\\ x+arctg\ , \text{якщо } x>1\\ \frac{x^2}{x+\frac{x}{x^3+1}}\ , \text{якщо } x=0{,}5\end{cases}$ |
| 20 | $y=\begin{cases}1+\dfrac{x^2}{2}+\dfrac{x^3}{4}\ , \text{якщо } x<0\\ xarctgx\ , \text{якщо } x>5\\ \dfrac{\ln(x+0{,}1)}{\sin(x+0{,}2)}\ , \text{якщо } 1<x<1{,}5\end{cases}$ |
| 21 | $y=\begin{cases}sin^2(x+5)\ , \text{ якщо } x<0\\ \sqrt[3]{x}+e^x\ , \text{ якщо } x>1\\ \dfrac{cos^2x+sin^2x}{ctgx^2}\ , \text{в інших випадках}\end{cases}$ |
| 22 | $y=\begin{cases}\sqrt[3]{\dfrac{x}{100}}+sin^2x^2\ , \text{ якщо } x<0\\ \lvert 100-x\rvert cosx\ , \text{ якщо } x>1\\ e^{x+1,1}+\sin x\ , \text{якщо } x=1 \text{ або } x=0{,}22\end{cases}$ |
| 23 | $y=\begin{cases}\sqrt[3]{1+\sqrt{\lvert x-10\rvert}}\ , \text{ якщо } x\le -2\\ tg^2\,x+1\ , \text{ якщо } x>1\\ 1+(x+\cos(x-1)) - \text{в інших випадках}\end{cases}$ |
| 24 | $y=\begin{cases}\sqrt{1+\lvert x\rvert}\ , \text{ якщо } x\le 0\\ \dfrac{1+3x}{2+\sqrt[3]{1+x}}\ , \text{ якщо } x>1\\ arctg\,\dfrac{3x}{2}\ , \text{якщо } x=0{,}33\end{cases}$ |
| 25 | $y=\begin{cases}\dfrac{\sqrt{1+\lvert x\rvert}}{2+\lvert x\rvert}\ , \text{ якщо } x\le 0\\ \dfrac{1+x}{2+cos^3x}\ , \text{ якщо } x>1\\ 4\sqrt{1+x^3}\ - \text{в інших випадках}\end{cases}$ |
| 26 | $y=\begin{cases}\dfrac{3+\sin x}{1+x^2}\ , \text{ якщо } x\le 0\\ 2x^2cos^2x\ , \text{ якщо } x>1\\ tg^2\,\dfrac{x}{2}+\cos(x+2)\ , \text{якщо } x=0{,}65\end{cases}$ |
| 27 | $y=\begin{cases}\sqrt{1+2x^3-sin^2x}\ , \text{ якщо } x\le 0\\ \dfrac{2+x}{\sqrt[3]{2+e^{-0{,}1x}}}\ , \text{ якщо } x>1\\ 2\cos(x+5), \text{ якщо } 0<x\le 1\end{cases}$ |

| | |
|---|---|
| 28 | $y = \begin{cases} \dfrac{x^3}{3+\dfrac{x^3}{5}}, & \text{якщо } x \le -1 \\ \sqrt{1+x^2}, & \text{якщо } 0 < x \le 1 \; x = 5 \text{ або } x = 10 \\ \dfrac{1+x}{1+\sqrt[3]{1+e^{-0,2x}}}, & \text{якщо } x = 5 \text{ або } x = 10 \end{cases}$ |
| 29 | $y = \begin{cases} \sqrt[3]{1+\sqrt{\lvert x-10\rvert}}, & \text{якщо } x \le -2 \\ tg^2 x + 1, & \text{якщо } x > 1 \\ 1 + (x + \cos(x-1)) & - \text{ в інших випадках} \end{cases}$ |
| 30 | $y = \begin{cases} ln^2 x^3 + \sqrt{x}, & \text{якщо } 3 < x < 6 \\ \dfrac{x}{3+\dfrac{x^2}{5+x}}, & \text{якщо } x > 10 \\ x \ln x + arctgx, & \text{якщо } 6 \le x \le 10 \end{cases}$ |

**Контрольні питання.**

4. Вкажіть область бачимості всіх змінних, які ви оголосили під час виконання практичної роботи?
5. Які властивості форми є унікальними (тобто існують лише для об’єктів типу TForm).
6. Дані якого типу містять властивості Caption чи Text для візуальних компонентів?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити знімок форми під час виконання.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №11
# Тема: Робота з таблицями

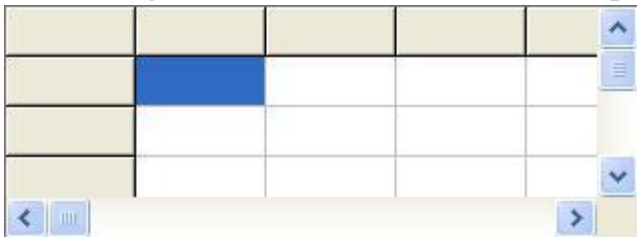
*Мета роботи*: вивчення оператора циклу. Придбання навичок роботи з таблицями.

**Теоретичні відомості.**

**Робота з таблицею StringGrid**

Компонент StringGrid знаходиться на сторінці Additional палітри компонентів.

StringGrid - компонент для відображення різних даних в табличній



формі. Як випливає з назви, комірки компонента StringGrid Delphi можуть містити дані, що мають тип String, а також відображати графіку.

Таблиця StringGrid складається з виділених сірим FixedCols і FixedRows - зафіксованих комірок- заголовків, і звичайних, білих клітинок. Вміст Fixed комірок заблоковано редагуванню, і міняється тільки програмно. За можливість редагування звичайних комірок відповідає одне із значень властивості Options.

Компонент StringGrid має можливість адресації кожної окремої комірки за номером стовпчика і рядка. Вміст комірки ( i , j ) , де де i - номер стовпчика, j - номер рядка, має вигляд:

*StringGrid1.Cells [ i , j ]*

і доступно як для читання, так і для запису. Тут, як і завжди, номери стовпців ( i ) та рядків ( j ) відраховуються від 0.

Виділена клітинка таблиці має

номер стовпчика : **StringGrid1.Col**

номер рядка: **StringGrid1.Row**

тому вміст виділеної комірки буде адресуватися так:

*S : = StringGrid1.Cells [ StringGrid1.Col , StringGrid1.Row ];*

Написання такого рядка - виснажливий процес. Тому користуйтеся оператором приєднання with :

```
with StringGrid1 do
 S : = Cells [ Col , Row ] ;
```

За багато властивості компоненту Delphi StringGrid відповідає властивість Options. У Інспектора Об'єктів Options - це список, що розкривається, який представляє собою елементи даної множини. Якщо значення елемента дорівнює True , то він присутній у множині , якщо False - то ні.

| Властивість | Значення |
|---|---|
| goFixedVertLine | Наявність вертикальних роздільних ліній між "фіксованими" комірками |
| goFixedHorzLine | Наявність горизонтальних роздільних ліній між "фіксованими" комірками |
| goVertLine | Наявність вертикальних роздільних ліній між "звичайними" комірками |
| goHorzLine | Наявність горизонтальних роздільних ліній між "звичайними" комірками |
| goRangeSelect | Можливість виділення діапазона комірок |
| goDrawFocusSelected | Зафарбовування комірки з фокусом вводу |
| goRowSizing | Можливість зміни висоти строчок мишкою |
| goColSizing | Можливість зміни широти стовпчиків мишкою |
| goRowMoving | Можливість змінювати номер строчки (тобто переміщувати її) мишкою |
| goColMoving | Можливість змінювати номер стовпчика (тобто переміщувати його) мишкою |
| goEditing | Можливість коригувати вміст комірки з клавіатури |
| goTabs | Порядок зміни фокусу<br>При значенні **True** фокус зміщується на наступну комірку в таблиці, **False** - на наснаступний компонент форми |
| goRowSelect | Виділяється вся строчка, що містить комірку, яка отримала фокус |
| goAlwaysShowEditor | При значенні **True** вміст комірки при отриманні фокусу відразу доступний редагуванню, **False** – спочатку потрібно клацнути на ній мишкою, або натиснути **Enter** чи **F2**<br>(прим.: не діє при **goRowSelect**=**True**) |
| goThumbTracking | При значенні **True** перемещення "бігунка" прокрутки мишкою викликає одночасне переміщення комірок, **False** – комірки переміщуються лише при відпусканні "бігунка" |

Як випливає з таблиці, за можливість редагувати вміст комірок з клавіатури відповідає елемент goEditing властивості-множини Options. У Інспекторі Об'єктів встановіть його значення в True.

Клітинок у таблиці, як правило, багато, і в рамках компоненту видно тільки частину з них. У програмі доступна інформація як про загальну кількість рядків і стовпців, так і номерах і кількості рядків і стовпців, видимих в рамках таблиці.

Кількість рядків у Delphi StringGrid дорівнює **StringGrid1.RowCount**.

Кількість стовпців у Delphi StringGrid дорівнює **StringGrid1.ColCount**.

Якщо клітинки не поміщаються в таблиці, з'являються смуги прокрутки.

У таблиці StringGrid також є властивість і для управління розміром комірок. Для всіх комірок

**DefaultRowHeight** - висота рядків за замовчуванням

**DefaultColWidth** - ширина стовпців за умовчанням

Ці значення ширини і висоти беруть всі нові комірки. При необхідності індивідуально встановити ширину і висоту стовпців і рядків відповідно, користуємося властивостями:

**RowHeights** - масив, що містить висоти рядків. Тобто, наприклад, RowHeights [ 5] - висота рядка з індексом 5

**ColWidths** - масив, що містить ширини стовпців. Тобто, наприклад, ColWidths [ 5] - ширина рядка з номером 5

Всі ці властивості налаштовуємо в обробнику події OnCreate Форми, так само як і написи заголовків, що розташовуються в рядках і стовпцях "фіксованої" зони таблиці. В результаті таблиця з'являється вже в "налаштованому" вигляді.

**Індивідуальні завдання.**

Створити додаток у середовищі програмування Delphi для обчислення значення функції y (згідно варіанту) в точках заданого інтервалу з заданим кроком обчислення. Значення змінних x та y вивести за допомогою об’єкта StringGrid. Дані, що виводяться, округлити до тисячних. Візуальне оформлення додатку – на вибір студента.

| № варіанта | Вид функції | Первинні данні | | | | |
|---|---|---|---|---|---|---|
| | | a | b | $X_н$ | $X_к$ | h |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | $y=\frac{arctgbx}{1+sin^2 x}$ | - | 0,75 | 1,35 | 6,5 | 0,8 |
| 2 | $y=\sqrt[5]{\frac{a+bx}{ln^2 x}}$ | 19,6 | 7,8 | 14,6 | 34,8 | 6 |
| 3 | $y=\frac{a\,ln^2 x}{b+\sqrt{x}}$ | 1,38 | -1,2 | 60 | 100 | 10 |
| 4 | $y=\frac{sin^2 x}{\sqrt{x}+bx}$ | - | 1,68 | 1,2 | 2,4 | 0,2 |
| 5 | $y=\frac{ln^2 (x-b)}{a\sqrt{x}}$ | 0,36 | 5,5 | 10 | 50 | 6 |
| 6 | $y=\frac{e^{xa}+b}{1+cos^2 x}$ | 0,9 | 1,85 | 0 | 1,2 | 0,15 |
| 7 | $y=\frac{a+\sqrt[3]{x}}{sin^2 bx}$ | 1,24 | 0,67 | 10,2 | 12,4 | 0,43 |
| 8 | $y=\frac{a\sqrt{x}-bx}{ln^2 x}$ | 2,8 | 0,45 | 40 | 60 | 4,5 |
| 9 | $y=\frac{\sqrt{a}x-b}{lg^3 x}$ | 20,2 | 7,65 | 3,5 | 4 | 0,1 |
| 10 | $y=e^{-x^2}\frac{a+bx}{sin\,x}$ | 4,6 | 2,5 | 0,75 | 1,8 | 0,3 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **11** | $y=\frac{tg^2\,ax-b}{e^{ax}}$ | **0,55** | **0,78** | **4,2** | **5,8** | **0,25** |
| **12** | $y=\frac{arctgbx}{1+\sqrt[5]{ax}}$ | **7,38** | **0,3** | **9** | **12** | **0,35** |
| **13** | $y=\frac{sin^3\,ax}{ax+b}$ | **0,28** | **1,35** | **1,2** | **7,5** | **0,5** |
| **14** | $y=\frac{e^{-bx}}{b+cos^3\,ax}$ | **0,9** | **0,66** | **2,3** | **8,9** | **1,3** |
| **15** | $y=\frac{ln^2\sqrt{x}}{a\sqrt{x}}$ | **0,85** | **-** | **17,2** | **24,6** | **2** |
| **16** | $y=\frac{arctg(a^3\,x)}{\sqrt{a^3+x^3}}$ | **1,16** | **-** | **0,25** | **1,28** | **0,33** |
| **17** | $y=\frac{1+\sqrt{bx}}{sin^2\,ax}$ | **0,4** | **10,8** | **0,84** | **1,25** | **0,15** |
| **18** | $y=\frac{a-e^{bx}}{ln^2\,x}$ | **1,28** | **0,03** | **12,6** | **34,9** | **7,6** |
| **19** | $y=\frac{(a+bx)^{25}}{1+cos^3\,ax}$ | **0,25** | **0,68** | **11,6** | **15,8** | **0,6** |
| **20** | $y=\frac{b+sin^2\,ax}{e^{-x12}}$ | **1,6** | **1,24** | **0,2** | **1,4** | **0,35** |
| **21** | $y=\frac{sin^2\,x-a}{bx}$ | **1,8** | **0,34** | **6,44** | **9,1** | **0,25** |
| **22** | $y=\frac{atg^2x}{b+0,7x}$ | **0,44** | **2,28** | **6,5** | **7,3** | **0,12** |
| **23** | $y=\frac{ln(a^2-x)}{b\,sin^2\,x}$ | **3,2** | **0,45** | **0,6** | **1,5** | **0,2** |
| **24** | $y=\frac{a-\sqrt{bx}}{1+cos^2\,x}$ | **17,6** | **10,4** | **1,9** | **3,8** | **0,3** |
| **25** | $y=\frac{ln^2\lvert x+a\rvert}{(x+a)^2}$ | **8,24** | **-** | **14,9** | **24,8** | **1,5** |
| **26** | $y=\frac{\sqrt{alnx}}{1+tg^2\,bx^2}$ | **7,32** | **0,05** | **13,3** | **14,5** | **0,08** |
| **27** | $y=\frac{1+tg^2\,x/a}{b+e^{x/a}}$ | **4,1** | **0,05** | **1,25** | **3** | **0,3** |
| **28** | $y=\frac{e^{ax}+a^{e^x}}{\sqrt{1+sin^2\,x}}$ | **2** | **-** | **0,6** | **0,02** | **0,05** |
| **29** | $y=\frac{\sqrt{ax+b}}{ln^2\,x}$ | **1,35** | **0,98** | **7,5** | **26,6** | **4,2** |
| **30** | $y=\frac{sin^2(b^2+x^2)}{\sqrt[3]{b^2+x^2}}$ | **-** | **2,5** | **1,28** | **5,34** | **0,4** |

## Контрольні питання.

3. Які обчислення називаються циклічними?
4. Яким чином у таблицю можна додати графічний елемент до комірки?
5. Яким чином можна змінити вміст фіксованої комірки?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити знімок форми під час виконання.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №12
# Тема: Робота з текстовими даними

*Мета роботи*: вивчення оператора циклу. Придбання навичок роботи з зі строками.

**Теоретичні відомості.**

Компонент Memo перебуває на сторінці "Standard" палітри компонентів. Його основне призначення - робота з великою кількістю рядків (введення, відображення та редагування текстового матеріалу). Memo дозволяє вводити багатостроковий текст з клавіатури, завантажувати його з файлу, редагувати і зберігати у файл текстового формату.

Простота текстового редактора компонента Delphi Memo полягає в тому, що текстовий редактор Memo не володіє можливостями форматування тексту. Це означає, що всі атрибути вибраного шрифту будуть застосовуватися до всього тексту.

Текст в компоненті Delphi Memo розміщується порядково. Тому є доступ до кожного рядка тексту окремо. Рядки в редакторі Delphi Memo є об'єктами Lines [ i ] типу String, де i - номер рядка, нумерація від нуля. Об'єкт Lines [ i ] доступний і для читання, і для запису. Відповідно, текст у компоненті Memo можна редагувати не тільки з клавіатури, а й програмно.

Як і в багатьох інших текстових редакторах у компонента Memo є можливість використовувати загальноприйняті гарячі клавіші, такі як: Ctrl+X - виділений текст вирізається і поміщається в буфер обміну, Ctrl+C - копіюємо виділений текст в буфер обміну, Ctrl+V - вставляємо текстовий вміст з буфера обміну в місце знаходження курсору, для скасування останньої команди використовуємо Ctrl+Z.

Для створення нових рядків служать методи

**Add ()**

**Insert ()**

Метод Add () додає новий рядок в кінець, а метод Insert () впроваджує новий рядок перед зазначеним, для чого метод Insert () має відповідний параметр:

```
begin
  Memo1.Lines.Add ('остання строчка в полі');
  Memo1.Lines.Insert (2, 'третій рядок');
end;
```

Memo має також метод для видалення рядки:

*Memo1.Lines.Delete (i);* / / Видалення рядка з індексом і

Кількість рядків у компоненті Memo міститься у властивості Count:

*N: = Memo.Lines.Count*;

Memo має властивості для переміщення курсору і виділення тексту :

**SelStart**

**SelLenght**

**SelText**

- Властивість SelStart типу Integer задає номер символу , після якого стоятиме курсор ( відраховується від початку всього тексту);

- Властивість SelLenght типу Integer визначає кількість виділених символів;

- Властивість SelText типу String містить виділений текст.

Для того , щоб виділення тексту було помітно на екрані, небхідно, щоб компонент мав фокусом введення. Тобто, щоб раніше або користувач перемістив туди курсор клавішею Tab або клацнув мишкою, або був виконаний оператор Memo1.SetFocus.

Щоб визначити, де зараз знаходиться курсор, на якому рядку, і позицію в рядку. Ці параметри містяться у властивості CaretPos компонента Memo, яке має тип TPoint, тобто точка - запис з координатами X і Y:

*Memo1.CaretPos.X* //позиція курсору в рядку (на відміну від SelStart)

*Memo1.CaretPos.Y* // номер рядка де знаходиться курсор

Для збереження вмісту текстового поля Memo в файл використовується функція SaveToFile ('mytetxt.txt'), а для вилучення - LoadFromFile ('mytetxt.txt'), де mytetxt.txt - текстовий файл розташований в каталозі програми.

**Індивідуальні завдання.**

Виконайте завдання минулої практичної роботи змінивши завдання таким чином:

- Функція, значення констант не змінюється;
- Початкове та кінцеве значення змінної х задати за допомогою об’єктів типу ComboBox, в які ввести по 10 значень, можливість вводу нових даних користувачем під час виконання відключити;
- Крок зміни параметру h задати за допомогою об’єкта ScrollBar, задавши його мінімальне та максимальне значення на власний вибір.
- Значення змінних х та у вивести за допомогою об’єкта або Memo, або ListBox, або RichEdit. Дані, що виводяться, округлити до тисячних.
- Візуальне оформлення додатку – на вибір студента.

**Контрольні питання.**

1. Які обчислення називаються циклічними?
2. Яким чином у можна додати нову строчку до об’єктів Memo, ListBox та RichEdit?
3. Яким чином можна змінити вміст фіксованої комірки?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити знімок форми під час виконання.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №13

# Тема: Робота з меню

*Мета роботи*: вивчення методів роботи з головним та контекстним меню.

**Теоретичні відомості.**

Найважливішим елементом користувальницького інтерфейсу є меню. Вибір пунктів меню здійснюється за допомогою миші або клавіатури.

Розрізняють два типи меню:

- головне меню форми;
- контекстне меню форми або компонента.

Головне меню завжди одне і розташовується під заголовком форми. Вибір одного з пунктів головного меню викликає появу на екрані підлеглого меню зі списком вкладених пунктів. Будь-який пункт підлеглого меню може бути або командою, або містити інше підпорядковане меню , про що свідчить стрілка праворуч від пункту. Рівень вкладеності підлеглих меню практично не обмежений, але сучасне уявлення про хороший інтерфейс вимагає, щоб вкладеність була мінімальною.

Контекстних меню може бути багато і вони не мають постійного місця всередині форми. Такі меню не пов'язані з головним меню і з'являються лише за спеціальною вимогою з боку користувача, як правило, по клацанню правою кнопкою миші, коли вказівник миші наведений на потрібний елемент. Пункти контекстного меню можуть містити підлеглі меню. Контекстне меню прив'язується до конкретного елементу форми і ідеально підходить для розміщення команд, специфічних тільки цьому елементу. Оскільки доступ до команд контекстного меню можна отримати швидше, ніж до команд головного меню, використання контекстних меню робить для користувача інтерфейс більш зручним.

Для створення головного та контекстного меню у середовищі Delphi має два різних компоненти: MainMenu і PopupMenu. Заповнення цих компонентів пунктами меню відбувається однаково, але результат буде різним. У першому випадку ми отримаємо стандартний рядок головного меню, а в другому - вікно контекстного меню.

**Головне меню**

Відображення у формі головного меню (main menu) забезпечує компонент MainMenu, розташований у палітрі компонентів на вкладці Standard. Помістіть цей компонент на форму і дайте йому назву.

Значок компонента MainMenu, який ви бачите на формі, відображається лише на етапі розробки. Він потрібен для того, щоб ви могли швидко активізувати компонент і перейти до установки його властивостей. Однак компонент MainMenu є невізуальними і на етапі виконання програми

його значок не відображається. Користувач бачить результат роботи компонента - рядок меню.

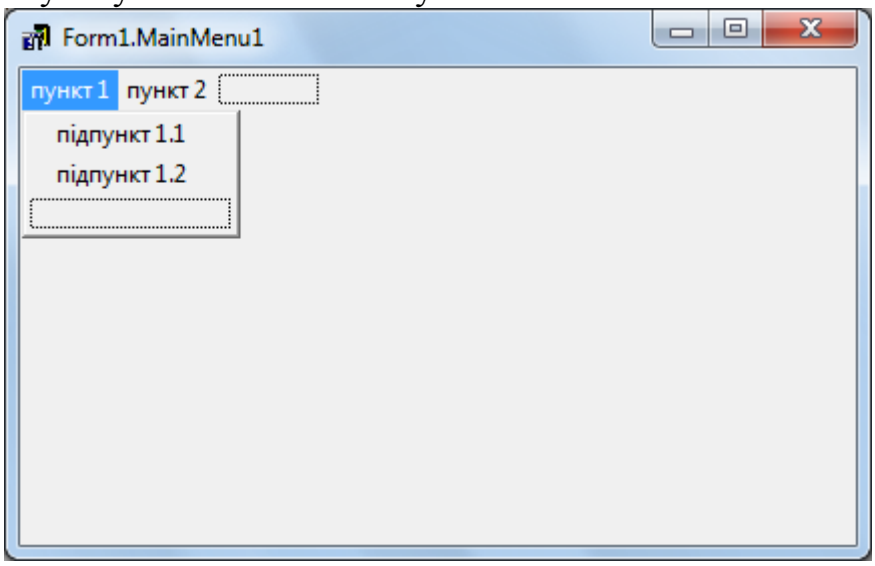
Основні властивості MainMenu:

| Властивість | Опис |
|---|---|
| AutoHotKeys | Значення maAutomatic позбавляє програміста від необхідності призначати пунктам меню "гарячі" клавіші (за допомогою спеціального символу & в тексті пунктів); компонент автоматично підбирає "гарячі" клавіші . Значення maManual вимагає , щоб " гарячі " клавіші призначив програміст |
| AutoLineReduction | Якщо дорівнює значенню maAutomatic , то при відображенні меню йдуть підряд пункти - роздільники малюються як один роздільник , а пункти - роздільники , що знаходяться на початку або кінці меню взагалі не показуються. Властивість AutoLineReduction застосовується при програмному додаванні і видаленні пунктів меню , щоб уникнути небажаних явищ на кшталт повторюваних і повислих розділових ліній. Якщо властивість AutoLineReduction дорівнює значенню maManual , то усі меню малюються як є. |
| AutoMerge | Визначає , зливається чи головне меню вторинної форми з головним меню головної форми . Спосіб злиття визначається значенням властивості GroupIndex кожного пункту меню верхнього рівня. |
| Images | Перелік піктограм , що відображаються поряд з пунктами меню. Властивість Images використовується спільно з властивістю ImageIndex компонентів MenuItem |
| Items | Масив пунктів меню. |
| OwnerDraw | Якщо дорівнює значенню True , то кожен пункт меню отримує можливість брати участь у процесі свого відображення за допомогою спеціальних подій OnMeasureItem і OnDrawItem . Подія OnMeasureItem відбувається в пункті меню , коли розраховуються розміри пункту. Подія OnDrawItem відбувається в пункті меню, коли пункт малюється на екрані. Якщо властивість OwnerDraw дорівнює значенню False , то пункти меню мають стандартний вигляд і події OnMeasureItem і OnDrawItem не відбуваються. |
| OnChange | Відбувається при зміні структури меню |

Виклик дизайнера меню здійснюється за допомогою команди Menu Designer, яка знаходиться в контекстному меню компонента MainMenu або подвійним клацанням мишею на значок меню.

Дизайнер меню працює в парі з вікном властивостей. Створення та видалення пунктів здійснюється в дизайнері меню, а властивості окремо

взятого пункту встановлюються у вікні властивостей.




Для створення меню користувач повинен ввести назви пунктів у відповідні поля редактора меню. Символ & забезпечує підкреслення наступного за ним символу при відображенні. Підкреслена буква використовується в комбінації з клавішею Alt для швидкого вибору пункту меню і називається гарячою клавішею. Зауважимо, що в деяких версіях операційної системи Windows «гарячі» клавіші підкреслюються тільки після натискання клавіші Alt.

Основні властивості пунктів меню:

| **Властивість** | **Опис** |
|---|---|
| Action | Задає так звану команду, яка буде виконуватися при виборі даного пунтках меню. |
| AutoCheck | Якщо дорівнює значенню True, то вибір пункту меню автоматично призводить до зміни значення властивості Checked на протилежне. Якщо дорівнює значенню False, то зміною властивості Checked управляє програміст. |
| AutoHotkeys | Значення maAutomatic позбавляє програміста від необхідності призначати пункту меню "гарячу" клавішу (за допомогою спеціального символу & у тексті пункту); компонент автоматично підбирає "гарячу" клавішу. Значення maManual вимагає, щоб "гарячу" клавішу призначив програміст. Значення maParent показує, що спосіб призначення гарячої клавіші визначається "батьківським" компонентом MainMenu. |
| AutoLineReduction | Якщо дорівнює значенню maAutomatic, то при відображенні меню поспіль йдуть пункти - роздільники малюються як один роздільник, а пункти - роздільники, що знаходяться на початку або кінці |

| | |
|---|---|
| | меню взагалі не показуються. Властивість AutoLineReduction застосовується при програмному додаванні і видаленні пунктів меню, щоб уникнути небажаних явищ на кшталт повторюваних і повислих розділових ліній. Якщо властивість AutoLineReduction дорівнює значенню maManual, то усі меню відображаються як є. Якщо властивість дорівнює значенню maParent, то спосіб визначається "батьківським" компонентом. |
| Bitmap | Значок, який відображається поруч з текстом пункту меню. Якщо для даного пункту меню зазначений індекс значка за допомогою властивості ImageIndex, то значення властивості Bitmap ігнорується. |
| Break | Якщо обрано mbBreak або mbBarBreak, то пункт меню починає новий стовпець. Значення mbBarBreak забезпечує відділення нового стовпця від попереднього вертикальної рискою. |
| Caption | Текст пункту меню. |
| Checked | Якщо дорівнює значенню True, то пункт меню містить мітку у вигляді "галочки" . |
| Default | Значення True говорить про те, що вибір пункту меню можна здійснити подвійним клацанням "батьківського" пункту меню. |
| Enabled | Визначає, чи доступний користувачеві даний пункт меню. |
| GroupIndex | Працює по-різному залежно від того, знаходиться пункт в підлеглому меню або в рядку головного меню. Пункти підлеглого меню з однаковим позитивним значенням GroupIndex погоджено перемикають між собою мітку - установка в одного пункту властивості Checked в значення True знімає позначку з іншого пункта. |
| Hint | Коротка підказка для користувача, яка відображається у рядку стану. |
| ImageIndex | Номер значка в списку Images компонента MainMenu. Значок відображається поруч з текстом пункту меню. Від'ємне значення властивості ImageIndex говорить про те, що для пункту меню значок не заданий. Властивість ImageIndex має пріоритет над властивістю Bitmap . |
| RadioItem | Якщо дорівнює значенню True, то мітка має вигляд жирної крапки. |
| ShortCut | Комбінація клавіш для виконання команди, не відкриваючи меню (гарячі клавіші). |

| | |
|---|---|
| SubMenuImages | Список піктограм, що відображаються поряд з пунктами підлеглого меню. Властивість SubMenuImages використовується спільно з властивістю ImageIndex компонентів MenuItem |
| Visible | Визначає, чи видно користувачеві пункт меню. |
| OnAdvancedDrawItem | Відбувається при малюванні окремо взятого пункту меню на екрані. Подія відбувається тільки в тому випадку, якщо відповідний компонент меню (MainMenu або PopupMenu) містить значення True у властивості OwnerDraw. Надає більш широкі можливості в порівнянні з подією OnDrawItem. |
| OnClick | Відбувається при виборі пункту меню користувачем. |
| OnDrawItem | Відбувається при малюванні окремо взятого пункту меню на екрані. Подія відбувається тільки в тому випадку, якщо відповідний компонент меню (MainMenu або PopupMenu) містить значення True у властивості OwnerDraw. |
| OnMeasureItem | Відбувається при розрахунку розмірів окремо взятого пункту меню перед його малюванням на екрані. Подія відбувається тільки в тому випадку, якщо відповідний компонент меню (MainMenu або PopupMenu) містить значення True у властивості OwnerDraw. |

Логічно пов'язані між собою команди прийнято відокремлювати від інших команд горизонтальною лінією. Для цього вставте новий пункт і запишіть у значенні властивості Caption символ мінуса (-).

Якщо ви хочете для підпункта меню створити підлегле меню, використовуйте при створенні комбінацію ‘Ctrl + →’

Для деяких пунктів меню призначають комбінації клавіш (shortcut), щоб виконувати команди, не відкриваючи меню. Вони прискорюють роботу з додатком і популярні серед досвідчених користувачів. Назви комбінацій клавіш відображаються праворуч від тексту відповідних пунктів. Наприклад , у багатьох програмах команді меню File / Open ... призначається комбінація клавіш Ctrl + O. Щоб призначити пункту комбінацію клавіш, активізуйте пункт у дизайнера меню , перейдіть до вікна властивостей і виберіть у списку значень властивості ShortCut необхідну комбінацію клавіш. Якщо її там немає, то введіть назву комбінації клавіш вручну.

**Контекстне меню**

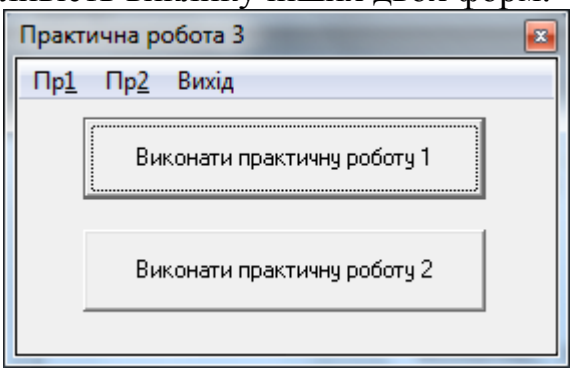
Контекстне (допоміжне) меню представлено у середовищі Delphi компонентом PopupMenu в палітрі компонентів на вкладці.

| Властивість | Опис |
|---|---|
| Alignment | Визначає місце появи меню щодо покажчика миші: paLeft - лівий верхній кут меню збігається з позицією курсору миші; paCenter - середина верхнього краю меню збігається з позицією курсору миші; paRight - правий верхній кут меню збігається з позицією курсору миші. |
| AutoHotkeys | Значення maAutomatic позбавляє програміста від необхідності призначати пунктам меню "гарячі" клавіші (за допомогою спеціального символу & в тексті пунктів); компонент автоматично підбирає "гарячі" клавіші . Значення maManual вимагає , щоб " гарячі " клавіші призначив. |
| AutoLineReduction | Якщо дорівнює значенню maAutomatic, то при відображенні меню поспіль йдуть пункти - роздільники малюються як один роздільник , а пункти - роздільники, що знаходяться на початку або кінці меню взагалі не показуються. Властивість AutoLineReduction застосовується при програмному додаванні і видаленні пунктів меню, щоб уникнути небажаних явищ на кшталт повторюваних і повислих розділових ліній. Якщо властивість AutoLineReduction дорівнює значенню maManual, то усі меню відображаються як є. |
| AutoPopup | Якщо дорівнює значенню True, то меню з'являється автоматично після натискання правої кнопки миші. Якщо дорівнює значенню False, то меню необхідно відображати програмно. |
| Images | Перелік піктограм, що відображаються поряд з пунктами меню. Властивість Images використовується спільно з властивістю ImageIndex компонентів MenuItem. |
| Items | Забезпечує нумерований доступ до пунктів меню. |
| MenuAnimation | Набір прапорців, що визначають спосіб появи меню на екрані: maLeftToRight - зліва направо, maRightToLeft - справа наліво, maTopToBottom - зверху вниз, maBottomToTop - знизу вгору, maNone - миттєве відображення. Щоб прапорці почали працювати , запустіть програму настройки екрану ( Start- > Settings - > Control Panel - > Display) і на вкладці Effects виберіть спосіб появи меню і підказок - Scroll Effect. |
| OwnerDraw | Якщо дорівнює значенню True, то кожен пункт меню отримує можливість брати участь у процесі свого відображення за допомогою спеціальних подій OnMeasureItem і OnDrawItem. Подія OnMeasureItem відбувається в пункті меню, коли розраховуються розміри пункту. Подія OnDrawItem відбувається в пункті |

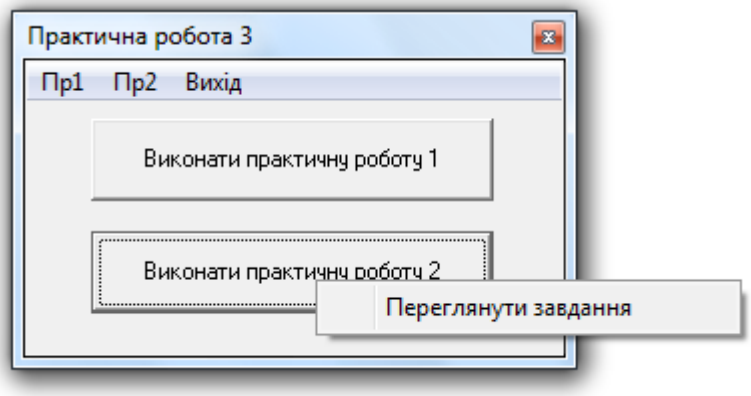
| | | |
|---|---|---|
| | **I** | меню, коли пункт малюється на екрані. Якщо властивість OwnerDraw дорівнює значенню False, то пукнту меню мають стандартний вигляд і події OnMeasureItem і OnDrawItem не відбуваються. |
| | TrackButton | Кнопка миші для вибору пункту меню: tbLeftButton - ліва кнопка, tbRightButton - ще й права кнопка. |
| | OnChange | Відбувається при зміні структури меню. |
| | OnPopup | Відбувається при виклику меню користувачем. |

**ндивідуальні завдання.**

Створити додаток у середовищі програмування Delphi, який буде складатися із 4 форм. Перша форма повинна бути заголовною та містити 2 кнопки та головне меню, за допомогою цих об'єктів повинна здійснюватися можливість виклику інших двох форм.




Під час натиснення на відповідну кнопку чи пункт меню додаток повинен викликати нову форму, на якій містяться результати виконання двох попередніх практичних робіт (форми 2 та 3). Створіть контекстне меню для кожної із кнопок, при виклику якого відкривається форма, у якій вказано завдання для відповідної практичної роботи.




Примітка: приблизний вигляд додатка наведено в завданні до практичної роботи, але студент має право змінювати оформлення, вигляд головного меню тощо. Бажано не переписувати завдання відповідних практичних робіт до нової програми, а просто вмонтувати відповідні файли до файлу проекта третьої роботи.

## Контрольні питання.

1. Що таке модальна форма?
2. Скільки об’єктів типу TMainMenu можна додати на форму?
3. Чи можна створювати контекстне меню для кожного об’єкта, розміщеного на формі?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити знімок форми під час виконання. У випадку, якщо студент просто додав існуючу форму, навести порядок дій, але сам вміст файлів можна не записувати.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №14
# Тема: Робота з перемикачами та групами перемикачів.

*Мета роботи*: придбання навичок роботи з перемикачами.

**Теоретичні відомості.**

**Компонент RadioButton**

Компонент RadioButton - це " радіокнопка ", і як випливає з назви, служить для " перемикання ". Це означає, якщо вибраний один з компонентів RadioButton, то з решти компонентів вибір автоматично знімається. Виходить, що в даний момент може бути вибраний тільки один з групи компонентів Delphi RadioButton.

Якщо в початковий момент жоден з компонентів не обраний, то досить зробити такий вибір - клацнути по одному з компонентів, і надалі можна тільки " перемикати " компоненти, зняти вибір вже можна лише програмно.

Компонент RadioButton складається з круглого віконця і текстового рядка. У віконці з'являється відмітка вибору даного компонента, текстовий рядок пояснює його зміст. За зміну текстового рядка відповідає властивість Caption. Основна властивість компонента Delphi RadioButton - Checked (тобто " вибрано " ) типу Boolean, доступна як для читання, так і для запису. Приклад:

```
RadioButton1.Checked = True ;
RadioButton2.Checked = False ;
```

Для вибору, наприклад, компонента RadioButton2 достатньо написати:

RadioButton2.Checked : = True ;// відмітка із RadioButton1 буде знята автоматично.

Для зняття вибору потрібно в програмі написати

RadioButton1.Checked : = False ;// тепер жоден з компонентів не вибраний. Вручну цього досягти неможливо.

Основною подією компонента Delphi RadioButton є, природно, OnClick, тобто клацання мишкою. У цей момент можуть бути виконані процедури, відповідні вибору цього компонента.

**Компонент CheckBox**

Компонент Delphi CheckBox це прапорець - незалежний перемикач. Прапорець Delphi CheckBox використовується в програмі для візуалізації станів включено - виключено. Кожне клацання мишкою по компоненту Delphi CheckBox змінює його стан на протилежний. Зміна стану перемикача

також доступна і в програмі.

CheckBox являє собою поєднання невеликого віконця, яке і візуалізує наявністю або відсутністю " галочки " стан компонента, і компонента Label - заголовка, пояснюючого призначення перемикача.

Основна властивість компонента CheckBox - Checked типу Boolean.

*CheckBox1.Checked = False ;*
*CheckBox1.Checked = True ;*

Властивість Checked доступно як для зчитування, так і для запису.

Головним чином, компонент Delphi CheckBox застосовується в умовних операторах, і допомагає сформувати умови вибору тієї чи іншої дії в програмі.

Компонент CheckBox - незалежний перемикач, тобто в групі з декількох компонентів кожен з них може бути встановлений в довільний стан, незалежний від стану інших компонентів групи ( на відміну від компонента Radiobutton ).

Крім властивості Checked, що дозволяє керувати станом компонента, у CheckBox є властивість State ( стан), яка може мати вже три значення . Перші два значення cbChecked ( " відзначено " ) і cbUnChecked ( " не відзначено " ), а третє - cbGrayed ( " не визначено" або "не знаю " ) стає доступним для вибору якщо встановити в True властивість AllowGrayed компонента. Властивості Checked і State пов'язані між собою:

якщо властивість State одно cbChecked , властивість Checked = True ;

якщо властивість State одно cbUnchecked або cbGrayed, властивість Checked = False.

При клацанні мишкою ці три стани змінюють один одного в наступній послідовності:

cbGrayed - не визначено (або "не знаю " ) ;

cbChecked - зазначено;

cbUnChecked - не відзначено ;

Розташуванням тексту в компоненті Delphi CheckBox також можна управляти. Для цієї мети служить властивість Alignment, що приймає значення:

*taRightJustify - розташування тексту справа ;*
*taLeftJustify - розташування тексту зліва.*

**Список CheckListBox**

Ще один компонент , що має індикатори - список CheckListBox. Це аналог компонента ListBox, але біля кожного рядка списку є індикатор, стан якого користувач може змінювати.

Стан індикаторів визначають дві властивості: State і Checked. Обидві ці властивості можна розглядати як індексовані масиви, кожен елемент якого відповідає індексу рядка. Ці властивості можна встановлювати програмно

або читати, визначаючи установки користувача. Наприклад, оператори:

*CheckListBox1.Checked [ 1 ]: = true ;*
*CheckListBox1.State [ 2 ]: = cbGrayed ;*

встановлюють індикатор другого рядка списку CheckListBox1 в стан обраного, а індикатор третього рядка - в проміжний стан (згадаймо, що індекси починаються з 0).

Оператор

```
for i : = 0 to CheckListBox1.Items.Count - 1 do
 if CheckListBox1.Checked [ i ] then ...
```

перевіряє стан всіх індикаторів списку, і для обраних користувачем рядків здійснює якісь дії (у наведеному операторі на місці цих дій просто поставлено три крапки ).

У компоненті CheckListBox мається також подія OnClickCheck , що виникає при кожній зміні користувачем стану індикатора. Його можна використовувати для обробки результатів зміни.

**Групування перемикачів**

**Групи радіокнопок - компоненти RadioGroup, GroupBox**

Радіокнопки утворюють групи взаємопов'язаних індикаторів, з яких зазвичай може бути вибраний тільки один. Вони використовуються для вибору користувачем однієї з декількох взаємовиключних

Панель RadioGroup може містити регулярно розташовані стовпцями і рядками радіокнопки. Напис в лівому верхньому куті панелі визначається властивістю Caption. А написи кнопок і їх кількість визначаються властивістю Items, які мають тип TStrings. Клацнувши на кнопці з трьома крапками близько цієї властивості у вікні Інспектора Об'єктів, ви потрапите в редактор списків рядків. У ньому ви можете занести написи, які хочете бачити близько кнопок, по одній у рядку. Скільки рядків ви запишете - стільки й буде кнопок.

Кнопки, що з'явилися в панелі після завдання значень Items, можна розмістити в декілька стовпців ( не більше 17 ), задавши властивість Columns. За замовчуванням Columns = 1, тобто кнопки розміщуються одна під одною.

Визначити, яку з кнопок вибрав користувач, можна по властивості ItemIndex, яка показує індекс обраної кнопки. Індекси, як завжди в Delphi, починаються з 0. За замовчуванням ItemIndex = -1, що означає відсутність обраної кнопки. Якщо ви хочете , щоб у момент початку виконання додатка якась із кнопок була обрана ( це практично завжди необхідно ), то треба встановити відповідне значення ItemIndex під час проектування.

Компонент RadioGroup дуже зручний, але не вільний від деяких недоліків. Його добре використовувати, якщо написи кнопок мають приблизно однакову довжину і якщо число кнопок в кожному стовпці (при

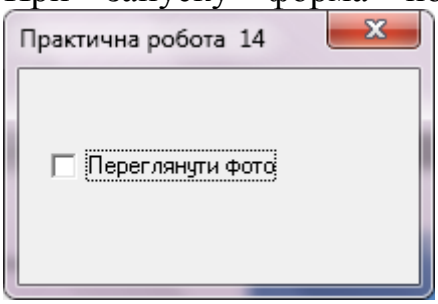
розміщенні їх у кількох стовпцях ) однакова. RadioGroup при розміщенні кнопок орієнтується на напис максимальної довжини. У подібних випадках бажано нерегулярне розташування кнопок. Таку можливість дають компоненти RadioButton, згруповані панеллю GroupBox. Панель GroupBox виглядає на формі так само, як RadioGroup, і напис в її верхньому лівому кутку також визначається властивістю Caption. Ця панель сама по собі порожня. Її призначення - служити контейнером для інших керуючих елементів. Окрема радіокнопка RadioButton особливого сенсу не має, хоча і може служити індикатором. Радіокнопки мають сенс, коли вони взаємодіють одна з однією в групі.

Радіокнопки RadioButton можуть розміщуватися не тільки в панелі GroupBox, але і в будь-якій панелі іншого типу, а також безпосередньо на формі. Група взаємопов'язаних кнопок в цих випадках визначається тим віконним компонентом , який містить кнопки. Зокрема , для радіокнопок, розміщених безпосередньо на формі, контейнером є сама форма. Таким чином , всі кнопки , розміщених безпосередньо на формі, працюють як єдина група, тобто тільки в одній з цих кнопок можна встановити значення Checked в true.

**Індивідуальні завдання.**

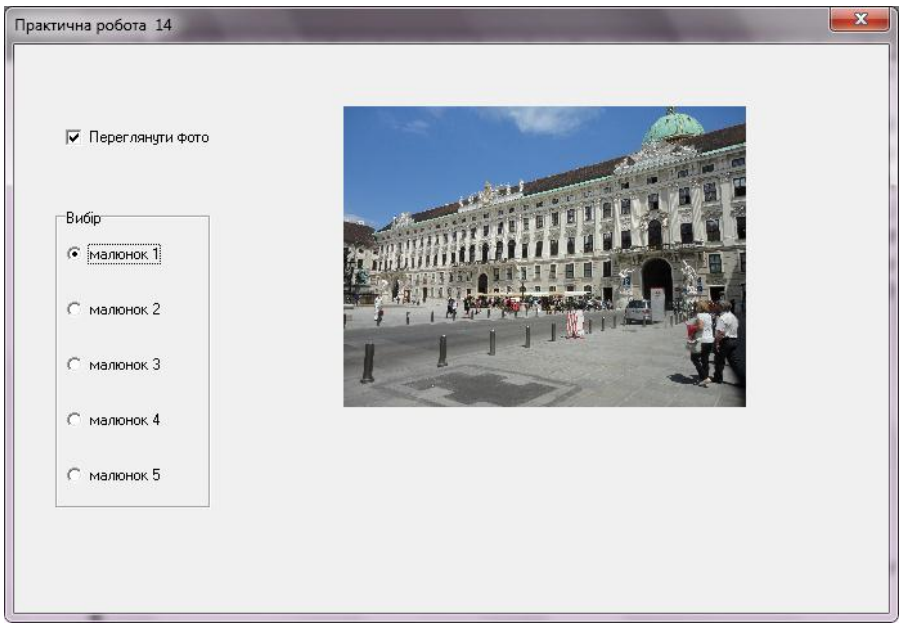
Побудувати оболонку наступного вигляду:

- На формі розмістити компоненти CheckBox, RadioButton1 – RadioButton5, компоненти ImageList, та Image;
- При запуску форма повинна мати вигляд, як на малюнку




- при встановленні прапорця розмір форми повинен змінитися, а також повинна з’явитися можливість перегляду у відповідному місці форми одного із зображень, список яких міститься у компоненті ImageList.
- при знятті прапорця форма повинна прийняти початковий вигляд (як при запуску).

Приблизний вигляд форми під час виконання - як на малюнку.




**Контрольні питання.**

1. Які стани можеть мати об’єкти «прапорець» та «радіокнопка»?
2. Яким чином можна реалізувати можливість вибору декількох радіокнопок, розміщених на формі?
3. Які компоненти можна використовувати для групування?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити знімок форми під час виконання.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №15

# Тема: Робота з файлами.

*Мета:* навчитися працювати з файлами.

**Теоретичні відомості.**

Потрібно згадати компоненти Delphi, які вміють працювати з файлами. Вони читають і зберігають свій вміст, рядки типу String, у файл текстового формату. Це компоненти ListBox, ComboBox і Memo, розташовані на першій же вкладці палітри компонентів.

Кожен рядок компонентів ListBox і ComboBox є об'єктом Items [i], а Memo - Lines [i], де і - номер рядка, який відраховується від нуля. Додавання рядків в компоненти виконується методами Add і Insert:

*Memo1.Lines.Add('Текст');*

*ComboBox1.Items.Add('Текст');*

Метод Add додає новий рядок в кінець. Метод Insert має додатковий параметр, який вказує, після якого рядка розмістити новий рядок. Доступ до рядків здійснюється так:

*ComboBox1.Items [0]: = 'Перший рядок';*

*ListBox1.Items [1]: = 'Другий рядок';*

Ну а тепер власне про збереження вмісту у файл. Для цього виконайте команду

*ListBox1.Items.SaveToFile ('Імя_файла.txt');*

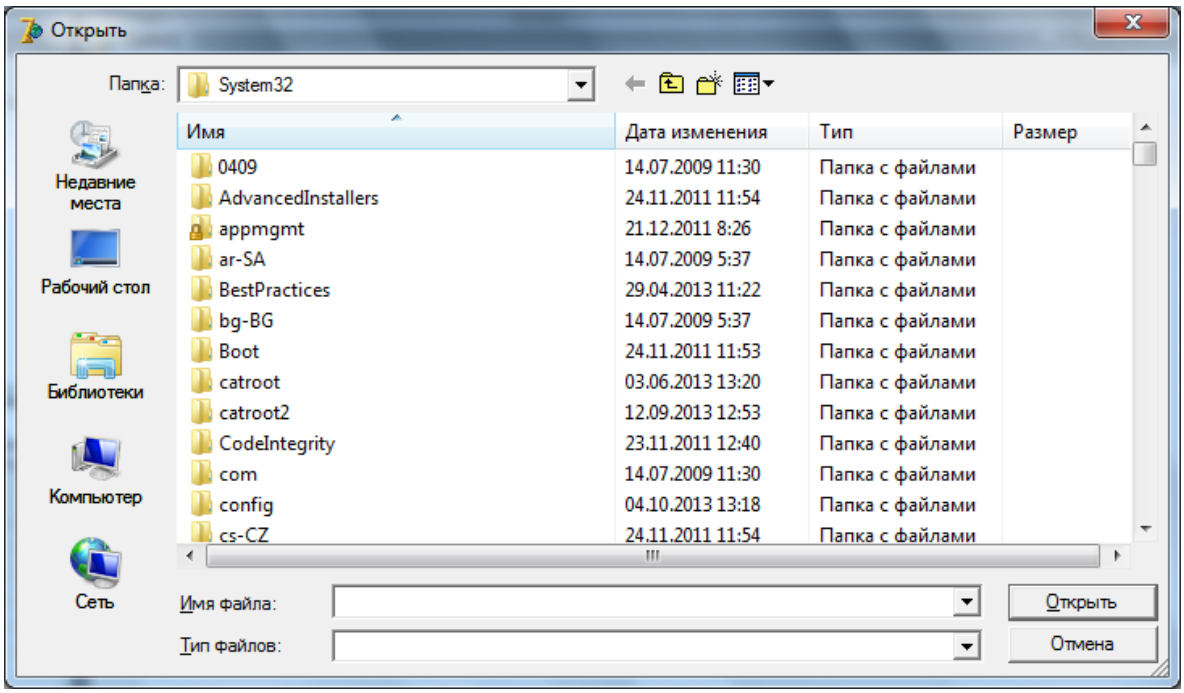
Розширення можна поставити будь-яке за бажанням.

Для завантаження служить метод LoadFromFile:

*ListBox1.Items.LoadFromFile ('Імя_файла.txt');*

Розглянемо компоненти, що дозволяють в працюючій програмі здійснювати вибір файлів. В Delphi діалоги вибору файлу дозволяють вказати програмі, з яким файлом ми хочемо працювати.

На вкладці палітри компонентів Dialogs знаходяться компонент Delphi **OpenDialog** і компонент Delphi **SaveDialog**. Всі Delphi діалоги, що знаходяться на цій вкладці, у тому числі і діалоги вибору файлу, невізуальні, тобто при перенесенні їх на Форму в працюючій програмі їх не видно, вони видно тільки на етапі конструювання. Компонент Delphi OpenDialog дозволяє відкрити в нашій програмі стандартне Windows- вікно діалогу відкриття файлу, компонент Delphi SaveDialog - вікно діалогу збереження.




Delphi-діалоги вибору файлу самі по собі нічого не роблять, а тільки надають налаштування, зроблені користувачем при виборі файлу. Найважливіший метод Delphi діалогів - Execute. Він спрацьовує в момент натискання кнопки "відкрити" або "зберегти" у вікні вибору файлу. Для прикладу давайте введемо в програму можливість вибору файлу для завантаження в редактор Memo, і збереження після редагування.

Для завантаження файлу пишемо команду в обробнику подій (для вибраного компоненту, наприклад, кнопки):

```
if OpenDialog1.Execute then
  Memo1.Lines.LoadFromFile ( OpenDialog1.FileName ) ;
```

У результаті вибору файлу властивість FileName компонента OpenDialog отримує значення повної адреси вибраного файлу, який ми і вставляємо в функцію завантаження файлу компонента Memo.

Якщо програма використовує кілька разів вираз OpenDialog1.FileName, то запис можна скоротити. У Delphi для такого випадку є так званий "оператор приєднання" with. Він використовується для будь-яких об'єктів, що мають довгий " хвіст " з властивостей , які доводиться записувати багато разів. Ось як він записується :

```
with Об'єкт do
  begin

  end ;
```

Властивості Об'єкта всередині логічних дужок begin / end можна записувати безпосередньо. Допускається перераховувати через кому кілька об'єктів. Природно, у разі , коли всередині дужок знаходиться один оператор, вони необов'язкові:

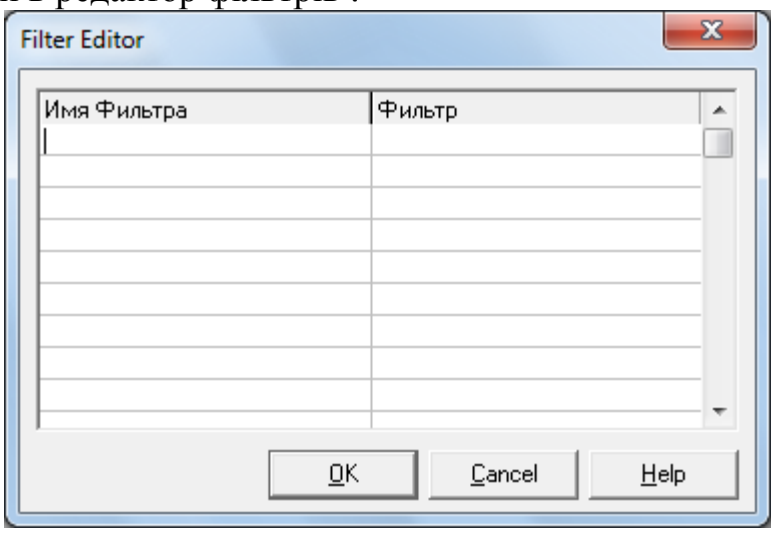
```
with OpenDialog1, Memo1 do
    if Execute then
      Lines.LoadFromFile (FileName);
```

Запис виходить більш компактним.

Так як властивості компонентів OpenDialog і SaveDialog однакові, збереження тексту виглядає абсолютно аналогічно:

*Memo1.Lines.SaveToFile(OpenDialog1.FileName);*

Для налаштування фільтрів використовуємо властивість Filter в наших компонентах. Налаштовується вона в інспекторі об'єктів. При виборі можна перейти в редактор фільтрів :




У колонці FilterName записуємо імена фільтрів, у колонці Filter - список масок файлів , розділених крапкою з комою. Маска файлу в даному випадку виглядає як

* . розширення_файла ;

Зірочка означає, що вибираються файли з будь-якими іменами, які підходять по розширенню.

Властивість Delphi діалогів Title дозволяє записати в заголовок потрібну нам фразу. Якщо залишити його порожнім, то в заголовку будуть стандартні "відкрити" або "зберегти".

Властивість InitialDir дозволяє в момент відкриття опинитися в потрібній нам директорії. Вона доступна як на етапі " конструювання ", так і програмно.

**Індивідуальні завдання:**

1) та 16) Із текстового файлу, обраного користувачем за допомогою об’єкта OpenDialog необхідно переписати вміст в інший текстовий файл з назвою, заданою користувачем, додаючи в початок кожного рядка її порядковий номер (десяткову запис цілого числа шириною в 5 символів) і пробіл. Вміст кожного з файлів вивести на форму за допомогою об’єкта Memo.

2) та 17) Із текстового файлу, обраного користувачем за допомогою об’єкта OpenDialog необхідно переписати в інший текстовий файл з назвою, заданою користувачем, всі його непусті рядки, які починаються і закінчуються однаковим символом. Вміст кожного з файлів вивести на форму за допомогою об’єкта Memo.

3) та 18) Із текстового файлу, обраного користувачем за допомогою об’єкта

OpenDialog необхідно переписати в інший текстовий файл з назвою, заданою користувачем, його вміст, крім рядків, що починаються з точки. Вміст кожного з файлів вивести на форму за допомогою об'єкта Memo.
4) та 19) Із текстового файлу, обраного користувачем за допомогою об'єкта OpenDialog знайти кількість порожніх рядків у ньому. Вміст заданого файлу вивести на екран за допомогою об'єкта Memo.
5) та 20) В текстовому файлі, обраному користувачем за допомогою об'єкта OpenDialog знайти довжину його максимальної рядка. Вміст файлу вивести на форму за допомогою об'єкта Memo. Найдовшу строчку та її довжину вивести за допомогою MessageDlg типу Information.
6) та 21) В текстовому файлі, обраному користувачем за допомогою об'єкта OpenDialog знайти кількість рядків, що починаються з заданого символу. Вміст файлу та знайдені рядки вивести на форму за допомогою об'єктів Memo.
7) та 22) В текстовому файлі, обраному користувачем за допомогою об'єкта OpenDialog знайти кількість рядків, що закінчуються заданих символом. Вміст файлу та знайдені рядки вивести на форму за допомогою об'єктів Memo.
8) та 23) Із текстового файлу, обраного користувачем за допомогою об'єкта OpenDialog видалити усі рядки заданої довжини. Вміст початкового та отриманого файлів вивести на форму за допомогою об'єктів Memo.
3) та 18) В текстовому файлі, обраному користувачем за допомогою об'єкта OpenDialog знайти кількість літер «я». Вміст файлу вивести на форму за допомогою об'єкта Memo.
10) та 25) Із текстового файлу, обраного користувачем за допомогою об'єкта OpenDialog, переписати вміст в інший текстовий файл, вставляючи символи переводу рядка після кожної точки. Вміст кожного файлу вивести на форму за допомогою об'єкта Memo.
11) та 26) Із текстового файлу, обраного користувачем за допомогою об'єкта OpenDialog, переписати вміст в інший текстовий файл, доповнюючи праворуч всі його рядки, менші за довжиною 20 символів, пробілами праворуч до 20 символів. Вміст кожного файлу вивести на форму за допомогою об'єкта Memo.
12) та 27) Дано послідовність символів, що закінчується крапкою і вводиться користувачем з клавіатури. Необхідно дописати її в текстовий файл, вставляючи в неї символи переводу рядка через кожні 40 символів. Необхідно передбачити можливість вибору користувачем файлу, в який буде проводитися запис, а також можливість створення нового файлу з вказаною назвою. Вміст отриманого файлу вивести на форму за допомогою об'єкта Memo.
13) та 28) У текстовий файл необхідно записати наступну інформацію:
X
X X
X X X
X X X X

X X X X X
X X X X X X
X X X X X X X
……..
X X X X X X X X ... X (40 елементів)

Необхідно передбачити можливість вибору користувачем файлу, в який буде проводитися запис, а також можливість створення нового файлу з вказаною назвою. Вміст отриманого файлу вивести на форму за допомогою об'єкта Memo.

14) та 29) У текстовому файлі записана послідовність слів, розділених пробілами. Переписати в інший текстовий файл всі дані з першого файлу, крім слів, «мама». Вибір файлу здійснюється користувачем за допомогою об'єкта OpenDialog. Необхідно передбачити можливість вибору користувачем файлу, в який буде проводитися запис, а також можливість створення нового файлу з вказаною назвою. Вміст кожного файлу вивести на форму за допомогою об'єкта Memo.

15) та 30) У текстовому файлі записана послідовність слів, розділених пробілами. Перевірити, чи міститься у файлі задане слово. Вибір файлу здійснюється користувачем за допомогою об'єкта OpenDialog. Результати з відповідними коментарями дописати в вихідний файл. Вміст отриманого файлу вивести на форму за допомогою об'єкта Memo.

**Контрольні запитання:**

1. Якими стандартними функціями можна користуватися лише при обробці текстових файлів:?
2. Вкажіть символ закінчення строки в текстовому файлі?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити знімок форми під час виконання.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №16
# Тема: Виклик зовнішніх додатків.

*Мета роботи*: придбання навичок роботи з зовнішніми додатками за допомогою середовища Delphi.

**Теоретичні відомості.**

Існує дві найбільш часто вживаних функцій . WinExec і ShellExecute

**Функція WinExec**

WinExec , залишена для сумісності з ранніми версіями Windows. У неї мало параметрів запуску.

Використовуваний модуль - Windows.

Опис:

*WinExec ( FileName : PChar ; CmdShow : Cardinal ): Cardinal ;*

де : FileName - шлях, назва програми, котру потрібно запустити, параметри командного рядка. Всі вказується в одному рядку;

CmdShow - стиль вікна. Показує, в якому стані буде відображатися вікно при запуску.

Параметри відображення вікна CmdShow:

| | |
|---|---|
| SW_HIDE | Додаток, що запускається робиться невидимим |
| SW_MAXIMIZE | Розширює вікно на весь екран |
| SW_MINIMIZE | Додаток, що запускається мінімізується. Після запуску активізується вікно вищого рівня, тобто вікно, звідки було запущено це додаток |
| SW_RESTORE | Робить вікно таким, яким воно було запущено в останній раз |
| SW_SHOW | Відображає вікно у своєму поточному розмірі і позиції |

Значення, що повертаються функцією WinExec:

| | |
|---|---|
| 31 | Нормальний запуск |
| 0 | Системі не вистачає пам’яті або ресурсів |
| ERROR_BAD_FORMAT | EXE файл пошкоджений або має неправильний формат (Windows на такий файл говорить, що він не являється додатком Win32) |
| ERROR_FILE_NOT_FOUND | файл не знайдено |
| ERROR_PATH_NOT_FOUND | Шлях до файлу задано не вірно |

Якщо ім'я виконуваного файлу в параметрі FileName не містить шляху директорії, Windows робить пошук виконуваного файлу в такій послідовності:

1. Каталог з якого додаток було запущено. Це робочий каталог

2. Системний каталог Windows (зазвичай C: \ WINDOWS \ SYSTEM).

3. Каталог Windows.

4. Директорії, зазначені у змінній операційного середовища в середовищі PATH. Дізнатися про каталогах цієї змінної можна, ввівши PATH в командному рядку сеансу MS-DOS. Зазвичай це директорії:

*C:\Inprise\vbroker\bin;*
*C:\Program Files\Borland\Delphi7\Bin;*
*C:\Program Files\Borland\Delphi7\Projects\Bpl\;*
*C:\WINDOWS\system32;*
*C:\WINDOWS;*
*C:\WINDOWS\System32\Wbem*

**Функція ShellExecute**

Функція ShellExecute не тільки запускає програми, а відкриває, редагує або друкує файл, з урахуванням зареєстрованої типу, а також відкриває зазначену папку провідником. Повертає Handle посилання на відкрите вікно.

Використовуваний модуль - ShellAPI . Його потрібно не забути вказати в розділі Uses:

*uses*
*Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, ShellAPI;*

Опис:

*ShellExecute ( hWnd : HWND ; Operation : PChar ; FileName : PChar ; Parametrs : PChar ; Directory : PChar ; CmdShow : Integer ) : HINST ;*

де: hWnd - Handle батьківського вікна (посилання на батьківське вікно), в який будуть передаватися повідомлення запущеної програми. Можна вказувати Handle - посилання на вікна вашої програми.

Operation – виконуєма операція. Open - відкрити , print - надрукувати , explore - відкрити папку. За замовчуванням open, якщо вказати nil.

FileName - ім'я файлу або документа, інтернет посилання, e - mail адреса.

Parametrs - параметри, що передаються додатком в командному рядку.

Directory - каталог за замовчуванням.

CmdShow - стиль вікна. Показує, в якому стані буде відображатися вікно при запуску.

| Стан | | Опис |
|---|---|---|
| SW_HIDE | 0 | Приховує вікно додатка і активує інше вікно. |
| SW_MAXIMIZE | 3 | Розгортає вказане вікно на весь екран. |
| SW_MINIMIZE | 6 | Згортає вказане вікно і активує наступне по порядку вікно. |

| | | |
|---|---|---|
| SW_RESTORE | 9 | Активує і показує вікно. Якщо вікно було згорнуте або розгорнуте на весь екран, Windows відновлює вікно до його нормальних розмірів і позиції. Додаток повинен вказувати цей прапор коли відновлює згорнуте вікно . |
| SW_SHOW | 5 | Активує вікно і відображає його в поточному розмірі та позиції. |
| SW_SHOWMAXIMIZED | 3 | Активує вікно і відображає його в розгорнутому вигляді. |
| SW_SHOWMINIMIZED | 2 | Активує вікно і відображає його в згорнутому вигляді. |
| SW_SHOWMINNOACTIVE | 7 | Відображає вікно в згорнутому вигляді. Активне вікно залишається активним . |
| SW_SHOWNA | 8 | Відображає вікно в його поточному стані. Активне вікно залишається активним. |
| SW_SHOWNOACTIVATE | 4 | Відображає вікно в його самому останньому розмірі та позиції. Активне вікно залишається активним . |
| SW_SHOWNORMAL | 1 | Активує і відображає вікно. Якщо вікно було згорнуте або розгорнуте на весь екран, Windows відновлює його початкові розміри і позицію. Додаток повинен вказувати цей прапор коли вперше відображає своє вікно . |

Замість параметрів Operation , Parametrs і Directory можна ставити nil. Вони є не обов'язковими параметрами для запуску.

ShellExecute у разі успішного запуску повертає Handle вікна, у разі невдачі повертає код.

Приклади:

*ShellExecute(Handle, 'open', 'c:\Windows\notepad.exe', **nil**, **nil**, SW_SHOWNORMAL);* // запустити блокнот

*ShellExecute(Handle, 'open', 'c:\windows\notepad.exe', 'c:\text.txt', **nil**, SW_SHOWNORMAL);* // відкрити файл c:\text.txt в блокноті

*ShellExecute(Handle, 'open', 'c:\archive', **nil**, **nil**, SW_SHOWNORMAL);* //Показати вміст каталога c:\archive

*ShellExecute(Handle, 'open', 'c:\MyDocuments\Letter.doc', **nil**, **nil**, SW_SHOWNORMAL);* //відкрити файл згідно розширення

*ShellExecute(Handle, 'open', 'http://www.mydelphi.info', **nil**, **nil**, SW_SHOWNORMAL);* //перейти за посиланням

Відкриття файлів через OpenDialog:

*if OpenDialog1.Execute then*

*ShellExecute(Self.Handle, 'open', PChar(OpenDialog1.FileName), nil, nil, SW_SHOWNORMAL);*

*end;*

Одночасно може бути запущена більш ніж одна програма. Кожна програма, що працює в деякий момент часу, називається процесом. Кожна команда, яку ви запускаєте, породжує хоча б один процес. Є кілька системних процесів, запущених весь час і підтримуючих функціональність системи.

У кожного процесу є унікальний номер, званий process ID, або PID, і, як і у файлів, у кожного процесу є власник і група. Інформація про власника і групі процесу використовується для визначення того, які файли і пристрої можуть бути відкриті процесом з урахуванням прав на файли. Також у більшості процесів є батьківський процес. Наприклад, при запуску команд з оболонки, оболонка є процесом і будь-яка запущена команда також є процесом. Для кожного запущеного таким шляхом процесу оболонка буде батьківським процесом.

Вивести на екран Handle вікна та ID процеса (через Label)

*var*

*hWnd: THandle;* //змінна для збереження

*hProc: DWORD;*

*begin*

*ShellExecute(hWnd, 'open', 'c:\Windows\notepad.exe', nil, nil, SW_SHOWNORMAL);*

*Label1.Caption := 'Handle вікна: '+ IntToStr(hWnd) + #10#13 + ' ID процеса: ' + IntToStr(hProc);*

**Індивідуальні завдання.**

Побудувати оболонку для організації вибору користувачем наступних дій:

- запуску деякого файлу, обраного користувачем за допомогою діалогу відкриття;
- запуску деякого стандартного Windows-додатка, назву якого користувач вводить через деяке поле вводу (наприклад, запустити калькулятор за допомогою вводу в відповідне поле строки calc.exe);
- перегляду вмісту деякої директорії, обраної користувачем;
- *можливість знищення деякого процесу по його номеру.

Примітка: останній пункт відноситься до завдання підвищеної складності і виконання його бажане, але не необхідне для захисту роботи.

**Контрольні питання.**

1. Які функції запуску додатків вам відомі?
2. Чим відрізняється режим 'open' від режиму ‘explore’ при роботі з директоріями?
3. Яким чином можна переглянути ID запущеного процеса?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити знімок форми під час виконання.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №17
# Тема: Створення багатосторінкових додатків.

*Мета роботи*: навчитися працювати з MDI-додатками.

**Теоретичні відомості.**

Термін MDI (Multiple Document Interface) дослівно означає багатодокументний інтерфейс і описує додатки, здатні завантажити і використовувати одночасно кілька документів або об'єктів. Прикладом такого додатка може служити диспетчер файлів ( File Manager ).

Зазвичай MDI -додатки складаються мінімум з двох форм - батьківської та дочірньої. Властивість батьківської форми FormStyle встановлюється рівним fsMDIForm . Для дочірньої форми встановіть стиль fsMDIChild.

Батьківська форма служить контейнером, що містить дочірні форми, які укладені в клієнтську область і можуть переміщатися, змінювати розміри, мінімізувати або максимізуватися. У вашому додатку можуть бути дочірні форми різних типів, наприклад одна - для обробки зображень, а інша - для роботи з текстом.

**Створення форм**

У MDI - додатку, як правило, потрібно виводити кілька примірників класів форми. Оскільки кожна форма являє собою об'єкт, вона повинна бути створена перед використанням і звільнена, коли вона більше не потрібна. Delphi може робити це автоматично, а може надати цю роботу вам.

**Автоматичне створення форм**

За замовчуванням при запуску програми Delphi автоматично створює по одному примірнику кожного класу форм в проекті і звільняє їх при завершенні програми. Автоматичне створення обробляється генеруємим Delphi кодом в трьох місцях.

Перше - розділ інтерфейсу у файлі модуля форми.

```
type
  TForm1 = class ( TForm )
  private
  { Закриті оголошення . }
  public
  { Відкриті оголошення . }
  end ;
```

У цьому фрагменті коду оголошується клас TForm1.
Другим є місце , в якому описується змінна класу.

```
var Form1 : TForm1 ;
```

Тут описана змінна Form1, яка вказує на екземпляр класу TForm1 і

доступна з будь-якого модуля. Зазвичай вона використовується під час роботи програми для керування формою.

Третє місце знаходиться в початковому тексті проекту, доступ до якого можна отримати за допомогою меню View / Project Source. Цей код виглядає як:

```
Application.CreateForm ( TForm1 , Form1 ) ;
```

Процес видалення форм обробляється за допомогою концепції власників об'єктів: коли об'єкт знищується, автоматично знищуються всі об'єкти, якими він володіє. Створена описаним чином форма належить об'єкту Application і знищується при закритті програми.

**Динамічне створення форм**

Хоча автоматичне створення форм корисно при розробці SDI -додатків, при створенні MDI - додатку воно, як правило, неприйнятно.

Для створення нового екземпляра форми використовуйте конструктор Create класу форми. Наведений нижче код створює новий екземпляр TForm1 під час роботи програми і встановлює його властивість Caption рівним ' New Form '.

```
Form1 : = TForm1.Create (Application ) ;
Form1.Caption : = ' New Form ';
```

Конструктор Create отримує від вас як параметр нащадка TComponent, який і буде власником вашої форми. Звичайно як власник виступає Application, щоб всі форми були автоматично закриті по закінченні роботи програми. Ви можете також передати параметр Nil, створивши форму без власника (або яка володіє собою - як вам більше подобається), але тоді закривати і знищувати її доведеться вам. У разі виникнення необроблюваної помилки така форма залишиться в пам'яті.

У наведеному нижче коді Form1 вказує тільки на останню створену форму.

```
with TFormI.Create (Application ) do
  Caption : = ' New Form ';
```

**MDIChildren і MDIChildCount**

Властивість MDIChildren є масивом об'єктів TForm, що надають доступ до створених дочірніх форм. MDIChildCount повертає кількість елементів у масиві MDIChildren. Зазвичай це властивість використовується при виконанні якого-небудь дії над усіма відкритими дочірніми формами. Ось код згортання всіх дочірніх форм командою Minimize All .

```
procedure TFormI.mnuMinimizeAllClick ( Sender: TObject ) ;
var
```

```
    iCount : Integers ;
  begin
    for iCount : = MDIChildCount - 1 downto 0 do
      MDIChildren [ iCount ] . WindowState : = wsMinimized ;
  end ;
```

Якщо ви будете згортати вікна в порядку зростання елементів масиву, цикл буде працювати некоректно, так як після згортання кожного вікна масив MDIChildren оновлюється і перевпорядковується, і ви можете пропустити деякі елементи.

**TileMode**

Це - властивість перечислимого типу визначає, як батьківська форма розміщує дочірні при виклику методу Tile. Використовуються значення tbHorizontal ( за замовчуванням) і tbVertical для розміщення форм по горизонталі і вертикалі.

**WindowMenu**

Професійні MDI -додатки дозволяють активізувати необхідне дочірнє вікно, вибравши його зі списку в меню. Властивість WindowMenu визначає об'єкт TMenuItem, який Delphi буде використовувати для виведення списку доступних дочірніх форм.

Для виведення списку TMenuItem має бути меню верхнього рівня. Це меню має властивість Caption , рівне swindow .

**MDI - події TForm**

У MDI - додатку подія OnActivate запускається тільки при перемиканні між дочірніми формами. Якщо фокус введення передається з не MDI -форми в MDI - форму, генерується подія OnActivate батьківської форми, хоча її властивість Active ніколи і не встановлюється рівним True. Якби OnActivate генерувався тільки для дочірніх форм, не було б ніякої можливості дізнатися про перехід фокусу вводу від іншої програми.

**MDI - методи TForm**

Специфічні для MDI -форм методи перераховані нижче.

Arrangelcons вибудовує піктограми мінімізованих дочірніх форм в нижній частині батьківської форми.

Cascade розпорядженні дочірні форми каскадом, так що видно всі їх заголовки.

Next і Previous переходить від однієї дочірньої форми до іншої , як ніби ви натиснули <Ctrl+Tab> або <Ctrl+Shift+Tab>.

Tile вибудовує дочірні форми так, що вони не перекриваються.

Оскільки модуль посилається на тип TfrmMDIChild, що знаходиться в модулі MDIChild, після рядка implementation слід додати ще один рядок:

```
uses MDIChild;
```

В модулі MDIChild додайте після рядка implementation рядок.

*uses MDIParent;*

Після компіляції MDI-додаток створено.

**Індивідуальні завдання.**

Побудувати багатодокументний додаток, який дозволяє користувачеві виконувати наступні дії:

- відкривати, проглядати, редагувати та зберігати текстові документи;
- відкривати для перегляду файли зображень;
- створювати нові текстові документи.

Кожна дочірня форма повинна мати своє меню (головне чи контекстне – на вибір користувача), а також в заголовку містити прізвище розробника. Користувач повинен мати можливість зміни типу розміщення вікон.

**Контрольні питання.**

1. Назвіть різницю між SDI та MDI-додатками?
2. Скільки дочірніх форм може містити MDI-додаток?
3. Яким чином можна закрити дочірню форму?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити знімок форми під час виконання.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №18
# Тема: Робота з класом TCanvas.

*Мета роботи*: вивчення методів прорисовки об’єктів.

**Теоретичні відомості для практичних робіт № 18 та №19.**

Робота з графікою в Delphi передбачає звернення до канви - властивості Canvas компонентів. Canvas Delphi це полотно, який дозволяє програмісту мати доступ до кожної точки ( пикселу ), і відображати те, що потрібно. Звичайно, малювати попіксельно для роботи з графікою в Delphi не доводиться, система Delphi надає потужні засоби роботи з графікою, що полегшують задачу програміста.

У роботі з графікою в Delphi в розпорядженні програміста знаходяться канва (полотно - властивість Canvas Delphi компонентів), олівець (властивість Pen), кисть (властивість Brush ) того компонента або об'єкта, на якому передбачається малювати. У олівця Pen і кисті Brush можна міняти колір (властивість Color ) і стиль (властивість Style). Доступ до шрифтів надає властивість канви Font. Ці інструменти дозволяють відображати як текст, так і досить складні графіки математичного та інженерного змісту, а також малюнки. Крім цього, робота з графікою дозволяє використовувати в Delphi такі ресурси Windows як графічні і відеофайли.

Вивчимо основні прийоми малювання. Малювання на формі (як, втім, і на багатьох інших об'єктах) відбувається через контекст пристрою (полотна). Цей об'єкт з'являється у вигляді підказки після крапки при наборі програми.

До цього об'єкту ( Canvas ) ми можемо приписувати різні інші об'єкти, зокрема кисть ( TBrush ), перо ( TPen ) і шрифт ( TFont ). Крім того, на полотні ( Canvas ) ми можемо використовувати картинку ( TBitmap ).

Так як при малюванні нам постійно доведеться використовувати конструкції виду Form1.Canvas, то краще цю частину винести за дужки за допомогою with .

код:

```
with Form1.Canvas do
begin
...
end ;
```

Між begin і end ми якраз і будемо малювати. Куди помістити весь цей код, залежить від завдання. Можна написати його в обробнику натиснення кнопки або ще де-небудь, де вам треба. Ми ж помістимо його в обробник FormPaint для нашої форми. Логічно це зробити тому, що, зокрема, ця подія виникає і при створенні форми. Крім того, якщо вікно нашої форми буде закрито іншим вікном, а потім знову опиниться видимим, то код FormPaint також буде виконуватися, так що ми зупинимося саме на цьому обробнику.

Основна властивість такого об'єкта як Canvas Delphi - Pixels [ i , j ] типу TColor, тобто це двовимірний масив точок (пікселів), що задаються своїм кольором. Малювання на канві відбувається в момент присвоєння небудь точці канви заданого кольору. Кожному пикселу може бути присвоєний будь-який доступний для Windows колір. Наприклад, виконання оператора

*Form1.Canvas.Pixels[100, 100]:=RGB(255, 0, 0);*
*Form1.Canvas.Pixels [ 100 , 100 ]: = clRed ;*

призведе до малювання червоною точки з координатами [ 100 , 100 ] .
Дізнатися колір пікселя можна зворотним присвоєнням :

*Color : = Image1.Canvas.Pixels [ 100 , 100 ] ;*

Тип TColor визначений як довге ціле ( LongInt ). Його чотири байти містять інформацію про частки синього ( B ), зеленого (G), і червоного (R ) кольорів. Частка кожного кольору може мінятися від 0 до 255.

*Brush.Color:=RGB(255, 255, 0);* //Задаємо кисть жовтого кольору

Для стандартних кольорів в Delphi визначений набір текстових констант . Побачити його можна , відкривши в інспектор об'єктів властивість Color, наприклад, тієї ж Форми .

**Клас Олівець ( TPen )**

Олівець - властивість Pen класу TCanvas, призначена для прорисовки ліній.

| | |
|---|---|
| Color | Колір олівця |
| Mode | Режим малювання. Визначає, зокрема, спосіб комбінування свого кольору з поточним кольором полотна |
| Style | Стиль лінії визначає, чи буде вона суцільною або пунктирною. можливі значення: psSolid (суцільна лінія ) ; psDash ( пунктирна ) та інші |
| Width | Товщина лінії в пікселах |

Клас Кисть - властивість Brush класу TCanvas, призначена для заповнення суцільних областей клієнтської частини форми відповідно до заданих шаблоном. Крім властивостей Color і Style, що збігаються з аналогічними властивостями класу TPen, в клас TBrush додано нову властивість Bitmap, яке дозволяє заповняти область не тільки суцільним кольором або пунктирними лініями, але і заздалегідь підготовленим точковим зображенням .

Клас Шрифт - властивість Font класу TCanvas, служить оболонкою ресурсу Windows, визначального поточний шрифт. Містить безліч стандартних властивостей, що описують характеристики шрифту. Найбільш

важливі з них:

| | |
|---|---|
| Color | Колір |
| Charset | Набір символів, який визначається використовуваної кодуванням. |
| Height | Висота шрифту в пікселях. Реально ця висота обчислюється за спеціальною формулою і може приймати негативні значення. Замість даної властивості краще використовувати властивість Size |
| Name | Назва шрифту, під яким він зареєстрований в Windows, наприклад Times New Roman , Courier та інші |
| Pitch | Профіль шрифту , що визначає , чи буде відстань між символами фіксованим ( fpFixed ) або змінним ( fpVariable ) , як це має місце в шрифтах Courier і Times New Roman відповідно. Якщо для шрифту явно задано значення , що не відповідне реальному профілю, система Windows автоматично підбере шрифт, всі символи якого найбільш точно відповідають зазначеним параметрам |
| Size | Висота шрифту в пікселах |

**Індивідуальні завдання.**

Побудувати за допомогою методу класу TCanvas на формі зображення (на вибір студента), що містить в собі декілька об’єктів.

**Контрольні питання.**

1. Яким чином можна вивести на екран малюнок з зовнішнього файлу? Які формати малюнків підтримує середовище програмування?
2. Яким чином можна вивести у вказану область вікна текст з заданими параметрами шрифту, розміром тощо.
3. Яким чином можна запрограмувати зміну розміщення зображення на екрані?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити знімок форми під час виконання.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №19
# Тема: Додання анімаційних ефектів до об’єктів полотна.

*Мета роботи*: вивчення методів прорисовки об’єктів.

**Індивідуальні завдання.**

Відкрити файл, створений під час виконання практичної роботи №18. Створити процедуру для зміни положення частини об’єктів на формі (деякі об’єкти повинні рухатися, деякі залишатися статичними). Початок руху – на вмбір студента (або при запуску додатку, або після деякої події).

**Контрольні питання.**

1. Яким чином можна вивести на екран малюнок з зовнішнього файлу? Які формати малюнків підтримує середовище програмування?
2. Яким чином відбувається прорисовка?
3. Яким чином можна запрограмувати зміну розміщення зображення на екрані?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити декілька знімків форми під час виконання.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Практична робота №20
# Тема: Організація друку.

*Мета роботи*: придбання навичок по друку даних за допомогою додатків Delphi.

**Теоретичні відомості.**

**Друк за допомогою функцій файлового введення / виводу**

Один зі методів друку – використання функції AssignPrn. Принцип роботи цієї функції співпадає з роботою функції Assign. Тут ми будемо використовувати найпростіші функції введення / виводу, тільки зв'яжемо вихідний потік не з файлом чи деяким об’єктом, а з принтером. Розглянемо приклад друку тексту, що міститься в компоненті TEdit:

```
var P : TextFile ;
begin
AssignPrn (P) ;
Rewrite (P) ;
Writeln ( P , Edit1.text ) ;
CloseFile (P) ;
End ;
```

Тут ми оголошуємо змінну P типу TextFile. Процедура AssignPrn є різновидом процедури Assign. Вона налаштовує змінну P на порт принтера і дозволяє працювати з ним як з файлом. Rewrite відкриває порт для роботи, а WriteLn - виводить інформацію на друк. Важливо закрити порт принтера командою CloseFile.

Цей спосіб можна використовувати для роздруківки рядків списку або яких-небудь інших даних, по мірі їх надходження в програму.

**Друк текстів методом Print**

Для роботи з принтером потрібно підключити модуль Printers :

```
uses
  Windows , Messages, SysUtils , Variants , Classes , Graphics , Controls ,
Forms , Dialogs , StdCtrls , Printers ;
```

При роботі з деякими об’єктами користувач може використати стандартний метод Print. Наприклад, цей метод доступний для об’єкта TRichEdit, що дозволяє друкувати текст, що зберігається в цьому компоненті. Цей метод має тільки один параметр - рядок, яка при перегляді в Windows-черзі друку є ім'ям завдання. Наприклад:

```
RichEdit1.Print (' Друк вмісту  RichEdit1 ');
```

Слід зауважити, що печать відтворює всі особливості форматування тексту, так само відбувається автоматичне перенесення рядків і розбиття тексту на сторінки. При цьому довжина рядків ніяк не пов'язана з розмірами компонента RichEdit.

При роботі з принтером як з полотном для початку друку викликається метод BeginDoc, потім проводиться виведення документа, завершується друк викликом методу EndDoc :

```
Printer.BeginDoc ;
with Printer.Canvas do
  begin
... Друк документа ...
  end ;
Printer.EndDoc ;
```

Наведу приклади друку за допомогою об'єкту TPrinter тексту та зображення .

Друк тексту можна здійснити так :

```
Printer.BeginDoc ;
Printer.Canvas.TextOut ( 10,10 , 'Виведення тексту');
Printer.EndDoc ;
```

Якщо Ви хочете надрукувати зображення, що знаходиться, наприклад , в компоненті Image1, то код може бути таким:

```
Printer.BeginDoc ;
printer.Canvas.Draw(0,0,image1.Picture.Graphic);
Printer.EndDoc ;
```

Друк деякого об’єкту типу Canvas можна здійснити так (розміри задаються у пікселях):

```
Printer.BeginDoc ;
printer.canvas.Rectangle(50,50,100,100);
Printer.EndDoc ;
```

**Друк форм**

У форм в Delphi є метод Print, який друкує всю клієнтську область форми. При цьому смуга заголовка і смуга головного меню форми не друкуються.

```
form1.Print;
```

Розглянемо деякі властивості друку форм. Не забувайте, що ці параметри повинні бути встановлені до виклику процедури друку.

Властивість PrintScale визначає опції масштабування зображення форми при друку:

| | |
|---|---|
| poNone | Масштабування не використовується. Розмір зображення може змінюватися залежно від принтера |
| poPrintToFit | Робиться спроба надрукувати зображення форми того ж розміру, який видно на екрані |
| poProportional | Збільшує або зменшує розмір зображення, підганяючи його під розмір сторінки. |

Ширина і висота полотна принтера доступні через властивості Printer.PageWidth і Printer.PageHeight . Закінчити друк на одній сторінці і почати друкувати на інший можна за допомогою методу Printer.NewPage .

Властивості класу TPrinter

| властивість | Призначення |
|---|---|
| Aborted | Має значення True, якщо користувач перервав процес друку |
| Canvas | Область виведення графічної інформації для принтера |
| Capabilities | Налаштування режиму друку (орієнтація, число копій і так далі) |
| Copies | Кількість копій |
| Fonts | Список шрифтів, підтримуваних поточним принтером |
| Orientation | Орієнтація паперу: книжкова або альбомна |
| PageHeight | Висота друкованої сторінки в пікселах |
| PageNumber | Номер поточної друкованої сторінки |
| PageWidtfi | Ширина друкованої сторінки в пікселах |
| Printerindex | Номер принтера властивості Printers |
| Printers | Список назв всіх принтерів , доступних в системі |
| Printing | Має значення True , коли виконується друк |
| Title | Стандартний заголовок сторінки |

**Компоненти Друк та Налагодження принтера**

Ви повинні використовувати TPrintDialog, щоб показати користувачеві діалог вибору і конфігурації принтера перед друком.

Налаштування параметрів сторінки друку (TPrintDialog, TPrinterSetupDialog, TPageSetupDialog)

Компонент TPrinterSetupDialog, призначений для налаштування параметрів роботи принтера, не має оригінальних властивостей, тому що ці настройки істотно розрізняються для різних видів принтерів . На підставі цього компонента можна створювати свої власні компоненти для конкретних

принтерів .

Компонент TPrintDialog відображає стандартне вікно друку Windows. Можна задати різні параметри друку, які визначаються наступними властивостями.

Властивості класу TPageSetupDialog

| Властивість | Призначення |
|---|---|
| Copies | Число копій |
| FromPage | Номер сторінки, з якої почнеться друк |
| MaxPaде | Максимальне число сторінок, яке може бути надруковане |
| MinPage | Мінімальне число сторінок, яке може бути надруковане |
| Options | Додаткові параметри настройки |
| PrintRange | Вид діапазону друкованих сторінок документа. Можливі значення: prAUPages (всі сторінки ); prSelection (сторінки обраного фрагмента); prPageNums (сторінки з діапазону From Page / To Page ) |
| PrintToFile | Має значення True, якщо виведення має здійснюватися не на принтер, а в файл |

Компонент TPageSetupDialog дозволяє налаштувати характеристики друкованих сторінок. Вони задаються у властивостях MarginBottom, MarginLeft, Margin Right, MarginTop (нижня, ліва, права, верхня межі друку), PageHeight, PageWidth ( висота і ширина сторінки), а також у властивості Options, що описує додаткові параметри. Одиниці виміру розмірів задаються у властивості Units .

**Індивідуальні завдання.**

Створити додаток у середовищі програмування Delphi, який буде містити в собі наступні об’єкти:

- Компоненти RichEdit та Image;
- Кнопку для друку форми;
- Кнопку для друку деякого текстового файлу (вибір здійснюється з форми за допомогою діалогу відкриття файлу та перед друком виводиться у RichEdit);
- Кнопку для друку зображення з компоненту Image (розмір зображення повинен масштабуватися таким чином, щоб зображення виводилося по розміру сторінки).

Після вибору варіанту друку повинен бути організований діалог вибору кількості копій друку. Під час друку на кожній сторінці повинно помістити дані про студента, що виконав роботу.

Примітка: студент має право замість кнопок вибору варіантів друку використовувати перемикачі або побудувати меню, за допомогою якого буде здійснено вибір.

**Контрольні питання.**

1. Яким чином можна вивести на друк строку тексту?
2. Вкажіть всі відомі вам об’єкти, що мають властивість Print.
3. Яким чином можна організувати діалог налаштування властивостей принтерів?

**Оформлення звіту:**

- Вказати назву, номер, мету виконання та завдання на практичну роботу.
- Навести алгоритм всіх процедур та функцій.
- Навести програмний код та зробити знімок форми під час виконання. До звіту додати надруковані за допомогою додатка аркуші.
- Сформувати висновки та дати відповіді на контрольні запитання.

# Список рекомендованої літератури


1. Бейсік, фортран і паскаль / Г.М. Бурлак, Н.І. Кузьменко
2. Глинський Я. М. Основи інформатики та обчислювальної техніки. Паскаль
3. Глинський Я. та ін. Паскаль
4. Караванова Т. Основи алгоритмізації та програмування
5. Тулякова Н.О. Практикум програмування на мові Паскаль
6. Богданов В. Основи алгоритмізації та програмування. Посібник
7. Ковалюк Т.В. Основи програмування
8. Глушаков С.В. і ін. Програмування на Delphi 5.0
9. Кетков Ю. и др. Практика по программированию: Visual Basic, C++, Builder, Delphi.

# Зміст

**Пояснювальна записка ..... 3**

**Практична робота №1 Тема: Складення програм з використанням умовних операторів. ..... 4**

**Практична робота №2 Тема: Складення програм з використанням операторів циклу ..... 11**

**Практична робота №3 Тема: Ініціалізація масивів та робота з пам'яттю ..... 16**

**Практична робота №4 Тема: Робота з двовимірними масивами ..... 22**

**Практична робота №5 Тема: Робота з записами..... 26**

**Практична робота №6 Тема: Робота з типізованими файлами ..... 31**

**Практична робота №7 Тема: Робота з текстовими файлами. ..... 37**

**Практична робота №8 Тема: Робота з процедурами та функціями..... 39**

**Практична робота №9 Тема: Робота з графічними примітивами. ..... 43**

**Практична робота №10 Тема: Робота з компонентами сторінки Standard. ..... 48**

**Практична робота №11 Тема: Робота з таблицями..... 58**

**Практична робота №12 Тема: Робота з текстовими даними ..... 63**

**Практична робота №13 Тема: Робота з меню..... 66**

**Практична робота №14 Тема: Робота з перемикачами та групами перемикачів. ..... 74**

**Практична робота №15 Тема: Робота з файлами. ..... 79**

**Практична робота №16 Тема: Виклик зовнішніх додатків. ..... 84**

**Практична робота №17 Тема: Створення багатосторінкових додатків. . 89**

**Практична робота №18 Тема: Робота з класом TCanvas. ..... 93**

**Практична робота №19 Тема: Додання анімаційних ефектів до об'єктів полотна. ..... 96**

**Практична робота №20 Тема: Організація друку. ..... 97**

**Список рекомендованої літератури..... 101**