Міністерство освіти і науки України
Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних робіт
«Розробка програм у середовищі «Lazarus»
з дисципліни
«Інформаційні технології та програмування в ДВЗ»

для студентів спеціальності: 142 «Енергетичне машинобудування»

Харків 2020

Міністерство освіти і науки України
Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних робіт
«Розробка програм у середовищі «Lazarus»
з дисципліни
«Інформаційні технології та програмування в ДВЗ»

для студентів спеціальності: 142 «Енергетичне машинобудування»

Затверджено
редакційно-видавничою
радою університету,
протокол №_ від __.__.2020 р.

Харків
НТУ «ХПІ»
2020

Методичні вказівки до виконання практичних робіт «Розробка програм у середовищі «Lazarus» з дисципліни «Інформаційні технології та програмування в ДВЗ» для студентів спеціальності 142 «Енергетичне машинобудування» / Упоряд. О.Ю. Ліньков, – Харків: НТУ «ХПІ», 2020. – 52с.

Укладачі: О.Ю. ЛІНЬКОВ

Рецензент:

Кафедра двигунів внутрішнього згоряння

ВСТУП

Основним фундаментальним поняттям програмування є дія. *Дія* виконується суб'єктом чи машиною над деяким об'єктом, за зміною стану якого можна робити висновки про результат. Дія має скінчену тривалість, а результат, до якого вона приводить, повинен бути цілком визначений. Для можливості виконання певної дії вона повинна мати опис мовою, зрозумілою виконавцеві (процесору). Такий опис називається *інструкцією (командою, оператором)*. Будемо вважати, що виконавець одночасно здійснює не більше однієї дії. При цьому послідовність дій, що виконуються одна за одною, утворює *процес*. Вважається, що однакові послідовності дій, здійснювані процесором в різні часи, утворюють відмінні один від одного процеси.

Опис множини процесів, що приводить до конкретного результату, називається *алгоритмом*. Він записується мовою, зрозумілою процесору, і складається з інструкцій та вказівок про порядок їх виконання.

У програмуванні основним виконавцем виступає процесор. Алгоритм, записаний мовою, «зрозумілою» процесору, називається *програмою*. Програми для ЕОМ складаються за допомогою спеціальних систем позначень, названих *мовами програмування*. Їх умовно поділяють на мови низького та високого рівнів.

Однією з найбільш поширених мов програмування (поряд з такими мовами, як C++, Java, Python) вважається розроблена в 70-их роках XX-го сторіччя доцентом факультету інформатики Стенфордського університету Ніклаусом Віртом мова Pascal. Сучасний її розвиток в першу чергу здійснюється завдяки широкому колу користувачів та незалежним проектам. Найбільшим незалежним проектом вільного середовища розробки програмного забезпечення є Lazarus. Розробниками якого є: Cliff Baeseman, Shane Miller, Michael A. Hess та інші учасники проекту.

Слід зазначити, що мова Pascal розроблена з урахуванням принципів структурного програмування. Для структурованих програм характерні відповідність конструкції мови логічному мисленню людини, можливість написання програми практично необмеженого розміру, будь-якої складності й будь-якого призначення. Низькою є імовірність допущення помилок. З цього приводу використання Паскалю вважається доцільним засобом раціоналізації інженерного труда.

ЗАГАЛЬНІ ПОЛОЖЕННЯ

Lazarus - вільне середовище розробки програмного забезпечення з відкритим кодом, яке використовує компілятор Free Pascal (є можливість використовувати синтаксис: Pascal: Object Pascal, Turbo Pascal, Mac Pascal, Delphi) з додаванням Інтегрованого Середовища Розробки (IDE). Lazarus є багатоцільовим інструментом програмування, тобто на ньому можна створювати програми різних типів (консольні додатки, динамічно-підкачуємі бібліотеки, додатки з графічним інтерфейсом) для різних операційних систем: Linux, FreeBSD, Mac OS X, Microsoft Windows.

Lazarus містить в собі редактор коду, візуальний проектувальник форм, а також бібліотеку компонентів. Бібліотека візуальних Компонентів Lazarus (LCL) включає еквіваленти для більшості компонентів з VCL (форми, кнопки, текстові поля і інші), які використовуються для створення додатків з графічним інтерфейсом.

Програма що має графічний інтерфейс складається з декількох файлів які складають проект. До проекту програми з графічним інтерфейсом входять наступні файли:

*.pas – вміщує основний програмний код на мові Free Pascal;

*.lpr – вміщує програмний код головного модуля;

*.lfm – вміщує опис форми;

*.lpi – вміщує опис проекту.

Ці файли створюються автоматично при створенні нового проекту користувачем. При цьому у теці з проектом з’являються наступні файли:

- project1.exe (файл програми що можна запустити на виконання). Цей файл створюється у результаті компіляції проекту програми.
- project1.ico (файл з "іконкою" проекту - зображенням лапи гепарда, воно з’являється у верхньому лівому куту вікна програми).
- project1.lpi (файл з інформацією про проект). Саме цей файл слід запускати щоб відкрити проект в середовищі Lazarus.
- project1.lpr (похідний файл проекту). Запуск цього файлу призведе до запуску середовища Lazarus з проектом.
- project1.lps (Конфігурація проекту у вигляді xml-коду)
- project1.res (Файл ресурсів які використовуються у проекті)
- unit1.lfm (Файл форми. В ньому записані налаштування усіх компонентів, які використовуються в модулі. Редагувати цей файл

власноруч не слід, для редагування цих даних необхідно використовувати «Редактор форм»).

- unit1.pas (Програмний код модуля мовою FreePascal).

Файли з ім’ям project – це файли усього проекту в цілому, файли з ім’ям unit – це файли модулів.

Інтерфейс середовища складається з кількох вікон (рис. 1):

- головне вікно;
- інспектор об'єктів;
- вікно редактора коду;
- вікно форми;
- вікно повідомлень.

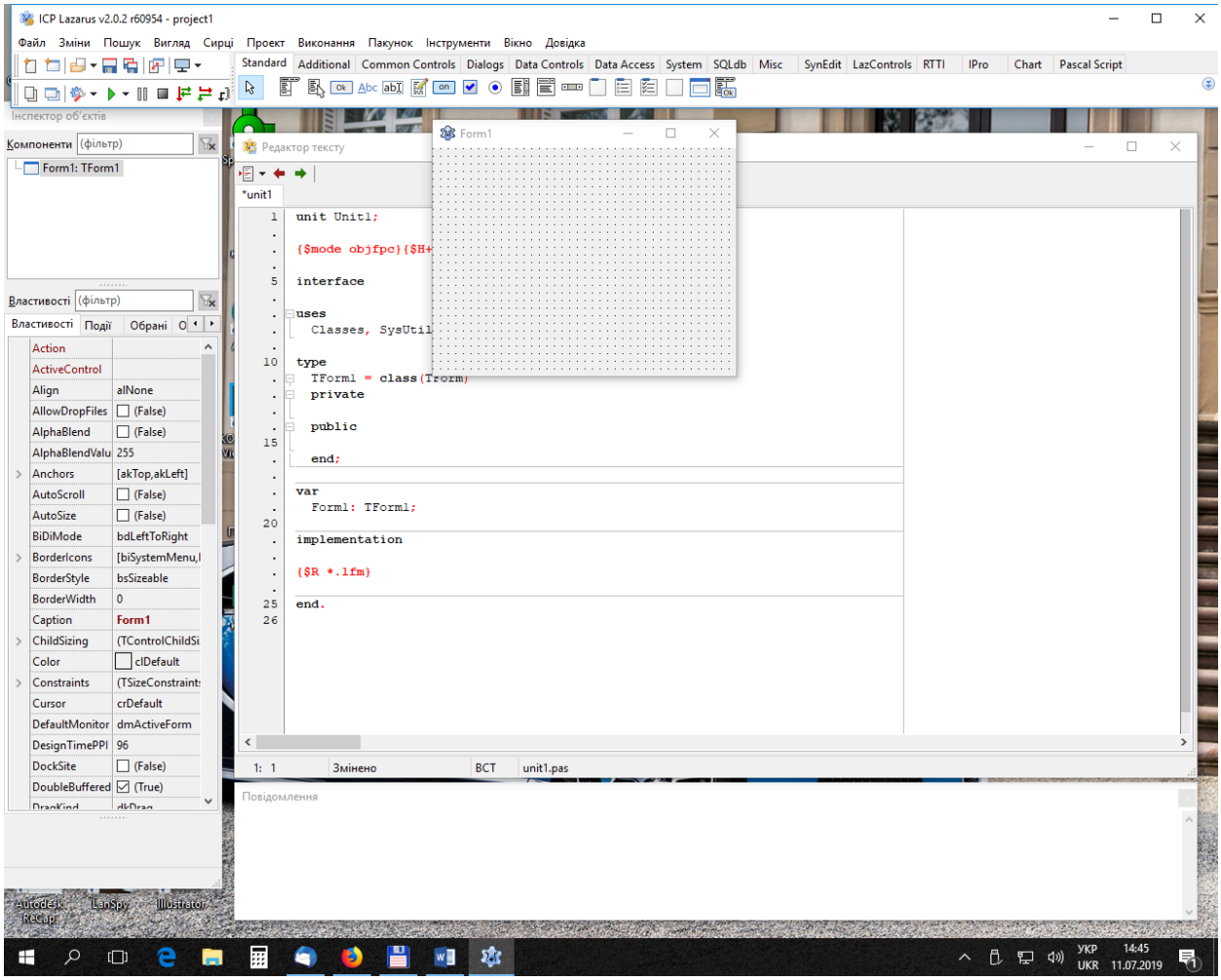


Рисунок 1 – Середовище Lazarus

За замовченням при запуску середовища створюється новий проект графічного додатку що використовує кросплатформену бібліотеку LCL для графічного інтерфейсу (або проект який був відкритий останнім, – в залежності від налаштувань).

Важливим елементом середовищ розробки програм є «компонент» – це віртуальний об’єкт, що має власні властивості (поля) та події. Набір доступних компонентів знаходиться на Палітрі компонентів (необхідні з них додаються на форму).

Таким чином, *засвоєння студентами знань і вмінь програмування у середовищі Lazarus з подальшим їх застосуванням в процесі навчання та в майбутній інженерній діяльності є основною метою курсу «Інформаційні технології та програмування в ДВЗ»*. Стосовно засвоєння цього курсу студентами спеціальності 142 «Енергетичне машинобудування» передбачено виконання циклу практичних робіт, в основу якого покладено розв’язання конкретних практичних задач.

ПРАКТИЧНА РОБОТА №1
ТЕМА: ОРГАНІЗАЦІЯ ВВЕДЕННЯ ТА ВИВЕДЕННЯ ДАНИХ

Мета: *засвоєння теоретичних знань та закріплення практичних навичок в розробці базових елементів програми з графічним інтерфейсом.*

1.1. Загальні положення

Виконання будь-якої програми на ЕОМ зводиться до переробки інформації. Усі дані характеризуються ім’ям, типом та значенням. Ім’я дозволяє ідентифікувати дані, тобто відрізняти їх від інших даних та обробляти саме їх. Тип задає множину можливих значень даних, спосіб їх зберігання, перетворення та використання. У кожний момент виконання програми значення визначає певний елемент цієї множини.

Дані поділяються на константи та змінні. Константи мають фіксовані й незмінювані значення. Значення змінних можуть змінюватися впродовж виконання алгоритму. На початку кожного процесу змінним, як правило, надаються певні значення, які далі змінюються дією присвоєння, як однією з фундаментальних дій програмування.

Кожна, навіть найпростіша програма, що реалізує визначений алгоритм, повинна мати:

- опис констант і змінних;
- інструкції з обробки інформації з присвоєнням результату змінним;
- інструкції з виводу необхідної інформації назовні.

Виконання роботи потребує попереднього вивчення таких питань теоретичної частини курсу:

1. Загальна структура програми.
2. Стандартні типи даних.
3. Організація лінійних алгоритмів.
4. Функції перетворення типів даних.
5. Оператор присвоєння.
6. Порядок обчислення значення арифметичного виразу.
7. Організація введення-виведення даних.

Ці питання розкривають сутність застосування обов’язкових елементів найпростішої програми.

1.2. Порядок виконання роботи

Задачею практичної роботи є розробка програми з лінійним алгоритмом. Програма повинна виконувати основні арифметичні дії над введеними користувачем числами:

A + *B* = *C*; *A* – *B* = *C*; *A* * *B* = *C*; *A* / *B* = *C*.

При підготовці к виконанню та під час проведення практичної роботи необхідно дотримуватись наступного плану:

1. Уяснити порядок дії програми. Програма повинна виконувати одну з чотирьох основні арифметичні дії над введеними користувачем двома числами.
2. Розробити алгоритм розв'язання задачі. Він складається з:

– Визначення констант і змінних.

Програма повинна мати три змінні (Табл. 1).

Таблиця 1 – Опис змінних

Параметр	Ім'я змінної	Тип даних
A – перше число	a	Real
B – друге число	b	Real
C – результат	c	Real

– Розробка форми вікна програми.

Вікно форми програми повинно містити такі компоненти:

Label – перший з інформацією про номер роботи, другий з інформацією про виконавця і третій з символом «=» (текст записується у полі Caption компоненту).

Edit – два компоненти до яких вводяться числа над якими будуть виконуватись арифметичні операції (у полі Text компонентів слід вказати початкове значення: 1).

Button – чотири кнопки з основними математичними операціями та кнопка завершення роботи програми (текст, який відображається на кнопках, записується у полі Caption компоненту).

Memo – до цього компоненту форми буде виводитись результат виконання обраної математичної операції (для очищення компоненту від надпису сліз звернутись до властивості Lines та викликати редактор рядків у якому очистити вміст).

Приклад розташування компонентів на формі показано на рис. 1.

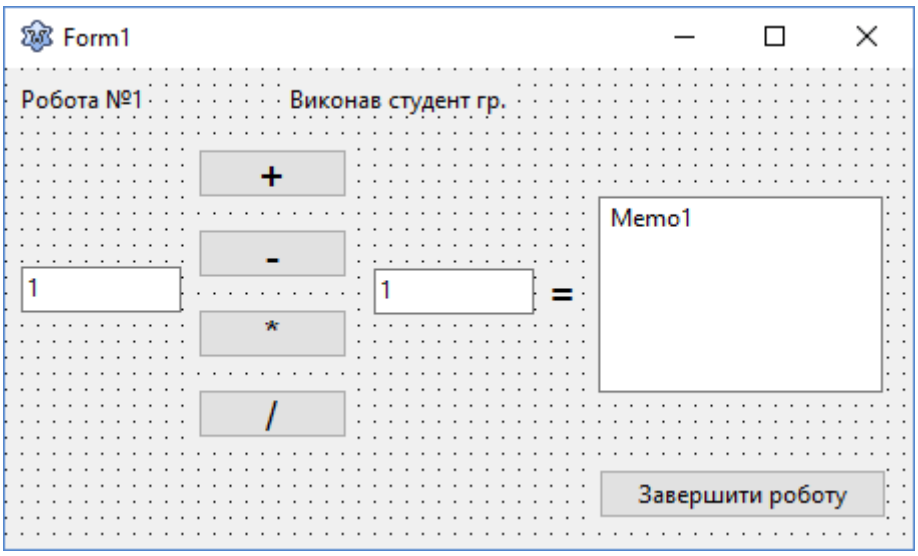


Рисунок 1 – Приклад вигляду форми вікна програми

– Розробка графічної блок-схеми алгоритму.

Для кожної з чотирьох кнопок записується алгоритм з відповідною математичною дією (рис. 2).

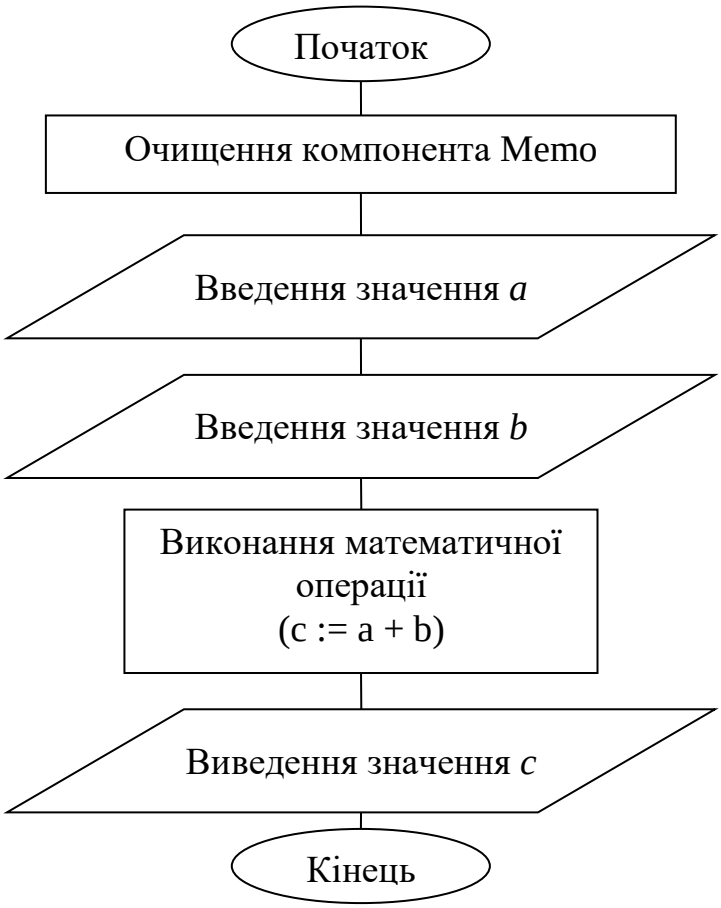


Рисунок 2 – Блок-схема алгоритму дії при натисканні на кнопку складання

3. Розробити текст програми відповідно до розробленого алгоритму.

При складанні програмного коду, з початку, слід вказати у блоці var імена змінних з табл. 1 та вказати їх тип даних:

```
var
    Form1: TForm1;
    a, b, c: Real;
```

Після опису змінних слід створити процедури які будуть виконуватись при натисканні на кнопку з відповідною арифметичною дією (для цього слід двічі кликнути на потрібній кнопці, після чого до тексту програмного коду буде автоматично додано опис нової процедури).

До створених процедур слід додати програмний код який відповідає діям вказаним у відповідній блок-схемі. У програмному коді використовуються такі оператори та функції:

:= – оператор привласнення;

StrToCurr() – функція перетворення типу даних з рядка до відповідного числового;

FloatToStr() – функція перетворення типу даних з числового до рядка;

Приклад програмного коду процедури натискання на кнопку з операцією складання:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    Memo1.Lines.Clear; //очищення вмісту компоненту
    a:=StrToCurr(Edit1.Text); //привласнення змінній значення яке
                              // знаходиться у полі Text компонента
                              // Edit1
    b:=StrToCurr(Edit2.Text); //привласнення змінній значення яке
                              // знаходиться у полі Text компонента
                              // Edit2
    c:=a+b;
    Memo1.Lines.Add(FloatToStr(c)); //виведення результату
end;
```

Подібно виглядають і процедури з іншими математичними операціями.

Процедура натискання на кнопку завершення роботи програми виглядає наступним чином:

```
procedure TForm1.Button5Click(Sender: TObject);
begin
    Form1.Close;
end;
```

4. До початку виконання практичної роботи блок-схему алгоритму та текст програмного коду слід затвердити у викладача.
5. Здійснити розв'язання поставленої задачі за допомогою середовища програмування «Lazarus». Після одержання результату узгодити його з викладачем. Виписати отримані результати роботи програми з екрану. Текст програми вивести на принтер. По закінченні роботи проект програми повинен зберігатися на диску ЕОМ у окремому каталозі. Рекомендоване ім'я проекту – LR1_iix де ii – порядковий номер прізвища студента по журналу, а **x** – індекс групи.
6. За результатами практичної роботи підготувати звіт, до якого обов'язково входить:

– Назва (тема) практичної роботи.
– Мета роботи.
– Індивідуальне завдання.
– Розробка алгоритму розв'язання задачі:
  - Вибір констант та змінних (таблиця ідентифікаторів).
  - Вигляд форми програми.
  - Графічна схема алгоритму (блок-схема).
– Текст програмного коду.
– Одержані результати та роздруківка тексту програми.
– Формування архіву програм (наводиться ім'я проекту програми, та адреса його зберігання).
– Висновки щодо досягнення мети роботи при розв'язанні наданого індивідуального завдання.

ПРАКТИЧНА РОБОТА №2
ТЕМА: ВИКОРИСТАННЯ ОПЕРАТОРУ УМОВИ

Мета: *засвоєння і закріплення теоретичних знань та отримання практичних навичок зі створення розгалужених алгоритмів та використання оператора умови.*

2.1. Загальні положення

Для реалізації розгалуження алгоритму використовується оператор умови

Синтаксис оператора:

If <Умова> Then <Дія1> Else <Дія2>;

У разі виконання умови буде виконано Дія1 (це може бути будь-який оператор, для виконання кількох операторів слід користуватись операторними дужками Begin … End;), якщо умова не виконується буде виконано Дія2.

Частину Else <Дія2> можна опускати тоді у разі не виконання умови ніяких дії відбуватися не буде, а програма перейде до наступного оператора.

Виконання роботи потребує попереднього вивчення таких питань теоретичної частини курсу:

1. Загальна структура програми.
2. Організація розгалужених алгоритмів.
3. Стандартні типи даних.
4. Функції перетворення типів даних.
5. Оператор присвоєння, оператор умови.
6. Порядок обчислення значень арифметичного виразу.
7. Організація введення-виведення даних.

2.2. Постановка задачі та порядок виконання роботи

В задачу практичної роботи входить розробка програми з розгалуженим алгоритмом для визначення яке число введено користувачем: парне чи непарне.

При підготовці до виконання та під час проведення практичної роботи необхідно дотримуватись наступного плану:

1. Уяснити порядок дії програми. Програма повинна сповіщати яке число ввів користувач – парне чи непарне.
2. Розробити алгоритм розв’язання задачі. Він складається з:

– Визначення констант і змінних.

Програма повинна мати одну змінну (Табл. 2).

Таблиця 2 – Опис змінних

Параметр	Ім’я змінної	Тип даних
A – ціле число яке вводить користувач	a	LongInt

– Розробка форми вікна програми.

Вікно форми програми повинно містити такі компоненти:

Label – перший з інформацією про номер роботи, другий з інформацією про виконавця (текст записується у полі Caption компоненту).

Edit – компонент до якого вводиться число над якими буде виконуватись операція (у полі Text компонентів слід вказати початкове значення: 1).

Button – кнопка завершення роботи програми (текст, який відображається на ній, записується у полі Caption компоненту).

Приклад розташування компонентів на формі показано на рис. 3.

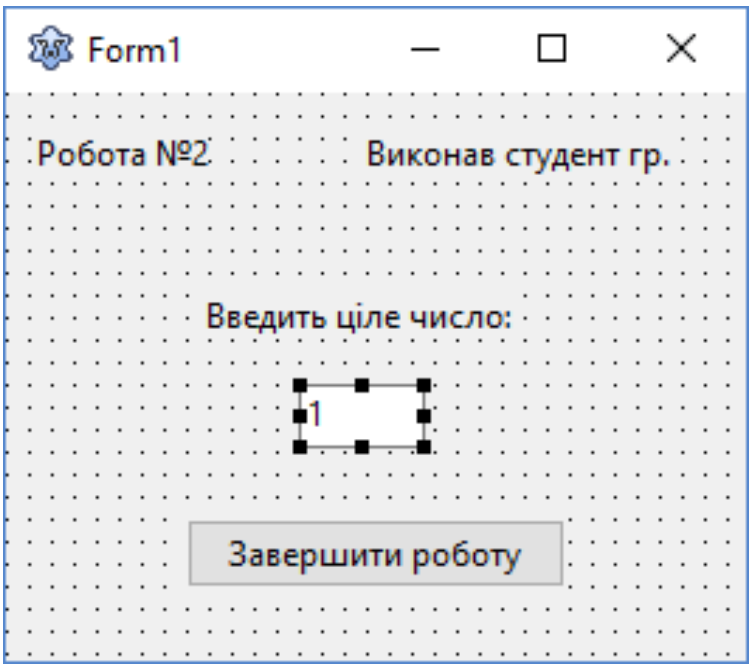


Рисунок 3 – Приклад вигляду форми вікна програми

– Розробка графічної блок-схеми алгоритму.

Алгоритм має розгалужену структуру (рис. 4).

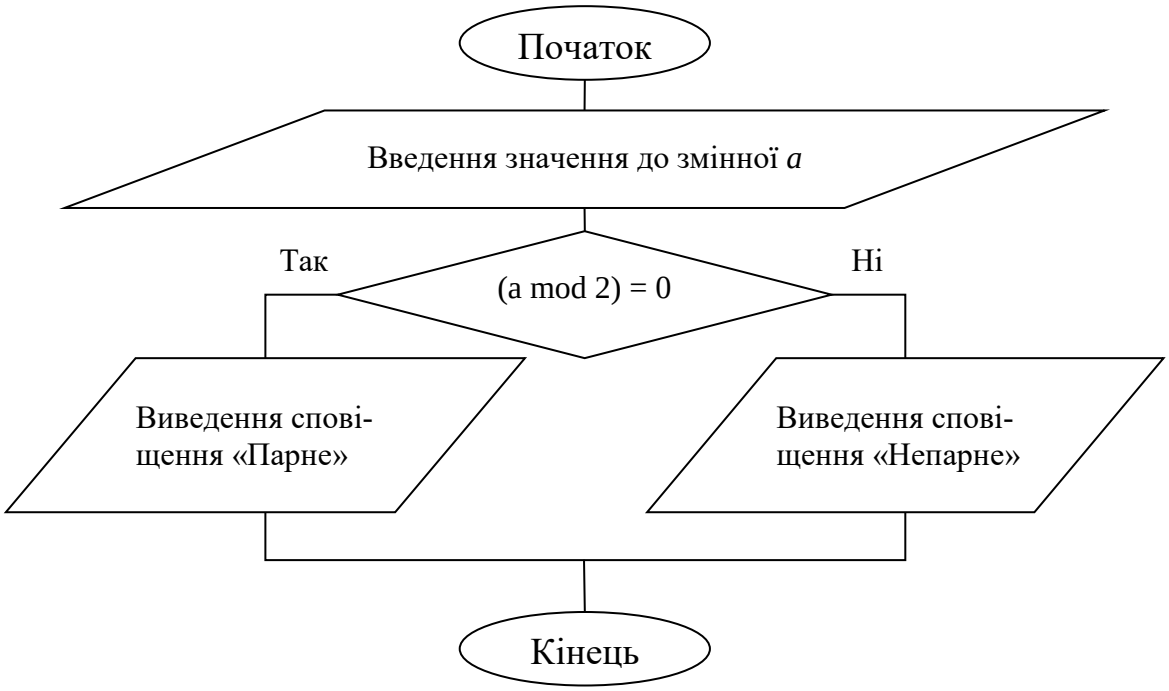


Рисунок 4 – Блок-схема алгоритму дії при введенні числа до компоненту Edit1

3. Розробити текст програми. При запису тексту додержуватись розробленого алгоритму.

При складанні програмного коду, з початку, слід вказати у блоці var імена змінних з табл. 2 та вказати їх тип даних:

```
var
  Form1: TForm1;
  a: LongInt;
```

Після опису змінних слід створити процедури які будуть виконуватись при введенні до компоненту Edit1 числа та натисканні на клавішу Enter (для цього слід обрати у списку подій компоненту Edit1 властивість OnEditingDone та встановити для неї нове значення Edit1EditingDone (натиснувши на три крапки), після чого до тексту програмного коду буде автоматично додано опис нової процедури) та при натисканні на кнопку завершення роботи програми (див. попередню роботу).

До створених процедур слід додати програмний код який відповідає діям вказаним у блок-схемі. У програмному коді використовуються такі оператори та функції:

:= – оператор привласнення;

mod – функція яка повертає залишок від операції ділення;

If <Умова> Then <Дія1> Else <Дія2>; – оператор умови;

StrToInt() – функція перетворення типу даних з рядка до цілочислового типу;

ShowMessage() – процедура виведення сповіщення.

```
procedure TForm1.Edit1EditingDone(Sender: TObject);
begin
  a:=StrToInt(Edit1.Text);
  If (a mod 2) = 0 Then ShowMessage('Парне') Else
Showmessage('Непарне');
end;
```

Процедура натискання на кнопку завершення роботи програми описана у першій роботі.

4. До початку виконання практичної роботи блок-схему алгоритму та текст програмного коду слід затвердити у викладача.

5. Здійснити розв’язання поставленої задачі за допомогою середовища програмування «Lazarus». Після одержання результату узгодити його з викладачем. Виписати отримані результати роботи програми з екрану. Текст програми вивести на принтер. По закінченні роботи проект програми повинен зберігатися на диску ЕОМ у окремому каталозі. Рекомендоване ім’я проекту – LR1_ix де і – порядковий номер прізвища студента по журналу, а **x** – індекс групи.

6. За результатами практичної роботи підготувати звіт, до якого обов’язково входить:

- Назва (тема) практичної роботи.
- Мета роботи.
- Індивідуальне завдання.
- Розробка алгоритму розв’язання задачі:
  - Вибір констант та змінних (таблиця ідентифікаторів).
  - Вигляд форми програми.
  - Графічна схема алгоритму (блок-схема).
- Текст програмного коду.
- Одержані результати та роздруківка тексту програми.

– Формування архіву програм (наводиться ім’я проекту програми, та адреса його зберігання).
– Висновки щодо досягнення мети роботи при розв’язанні наданого індивідуального завдання.

ПРАКТИЧНА РОБОТА №3
ТЕМА: ВИКОРИСТАННЯ ОПЕРАТОРУ ПОВТОРЕННЯ

Мета: *засвоєння і закріплення теоретичних знань та отримання практичних навичок з використання оператора повторення*.

3.1. Загальні положення

Оператор повторення виглядає наступним чином:

For <Параметр> := <Початкове значення> To <Кінцеве значення> Do <Дія>;

Виконання роботи потребує попереднього вивчення таких питань теоретичної частини курсу:

1. Загальна структура програми.
2. Стандартні типи даних.
3. Введення та виведення даних.
4. Функції перетворення типів даних.
5. Оператор присвоєння, оператор повторення.
6. Порядок обчислення значень арифметичного виразу.

3.2. Постановка задачі та порядок виконання роботи

В задачу практичної роботи входить розробка програми з використанням оператора повторення. Задачею є визначити суму чисел від 0 до числа заданого користувачем.

При підготовці до виконання та під час проведення практичної роботи необхідно дотримуватись наступного плану:

1. Уяснити порядок дії програми. Програма повинна вивести результат підсумовування чисел від 0 до числа яке задасть користувач.
2. Розробити алгоритм розв’язання задачі. Він складається з:
   – Визначення констант і змінних.

Програма повинна мати такі змінні (Табл. 3).

Таблиця 3 – Опис змінних

Параметр	Ім’я змінної	Тип даних
i – параметр циклу	i	Integer
a – число до якого слід провести підсумовування	a	Integer
y – результат	y	LongInt

– Розробка форми вікна програми.

Вікно форми програми повинно містити такі компоненти:

Label – надписи з інформацією про номер роботи, про виконавця, підпис компоненту введення кінцевого числа та компоненту з результатом розрахунку (текст записується у полі Caption компоненту).

Edit – компонент до якого вводиться значення кінцевого числа (у полі Text компоненту слід вказати початкове значення: 1).

Button – кнопка завершення роботи програми (текст, який відображається на ній, записується у полі Caption компоненту).

Memo – до цього компоненту форми буде виводитись результат виконання розрахунку (для очищення компоненту від надпису сліз звернутись до властивості Lines та викликати редактор рядків у якому очистити вміст).

Приклад розташування компонентів на формі показано на рис. 5.

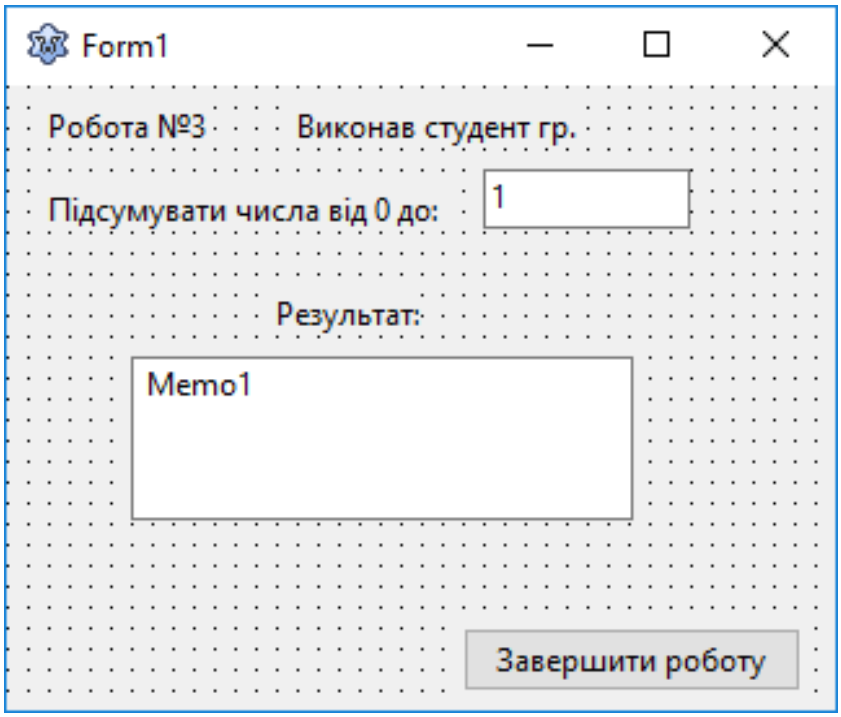


Рисунок 5 – Приклад вигляду форми вікна програми

– Розробка графічної блок-схеми алгоритму.

Алгоритм має розгалужену структуру (рис. 6).

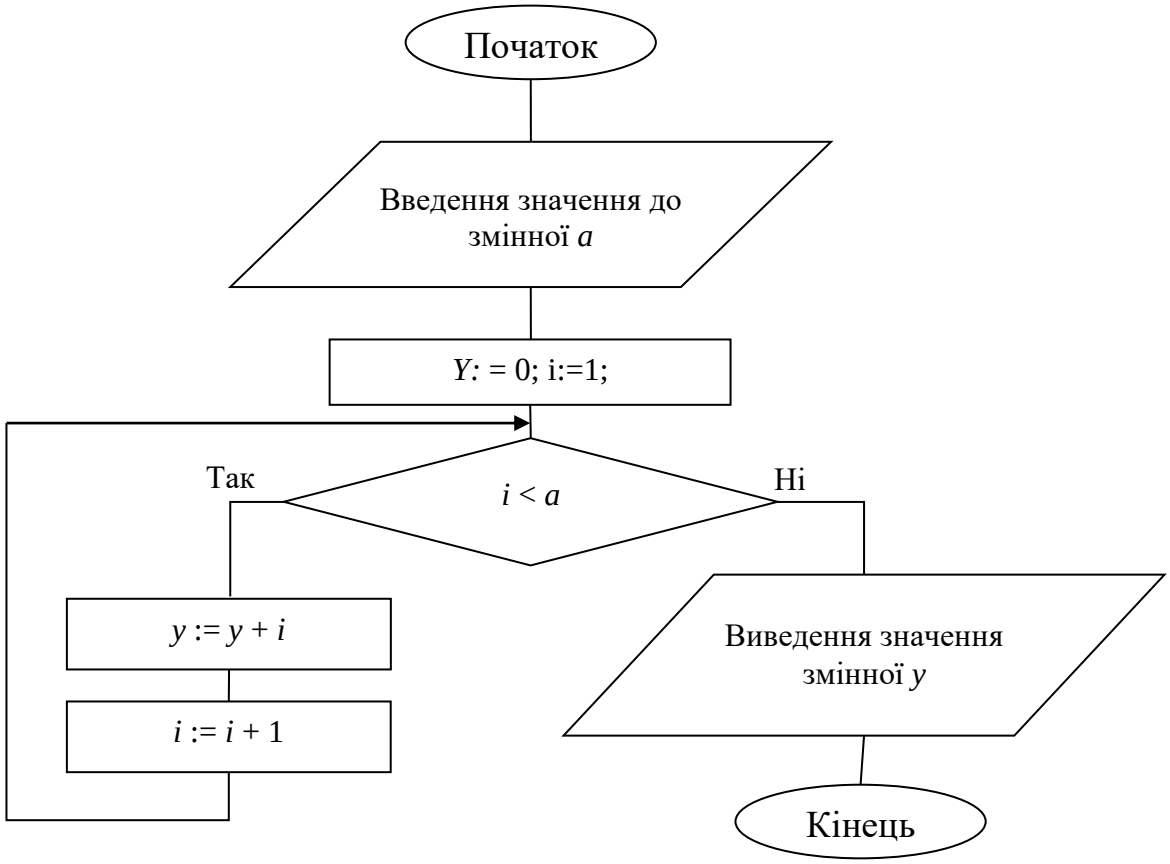


Рисунок 6 – Блок-схема алгоритму дії при введенні числа до компоненту Edit1

3. Розробити текст програми. При запису тексту додержуватись розробленого алгоритму.

При складанні програмного коду, з початку, слід вказати у блоці var імена змінних з табл. 3 та вказати їх тип даних:

```
var
  Form1: TForm1;
  a, i: Integer;
  y: LongInt;
```

Після опису змінних слід створити процедури які будуть виконуватись при введенні до компоненту Edit1 числа та натисканні на клавішу Enter (для цього слід обрати у списку подій компоненту Edit1 властивість OnEditingDone та встановити для неї нове значення

Edit1EditingDone (натиснувши на три крапки), після чого до тексту програмного коду буде автоматично додано опис нової процедури) та при натисканні на кнопку завершення роботи програми (див. попередню роботу).

До створених процедур слід додати програмний код який відповідає діям вказаним у блок-схемі. У програмному коді використовуються такі оператори та функції:

:= – оператор привласнення;

For <Параметр> := <Початкове значення> To <Кінцеве значення> Do <Дія>; – оператор повторення;

StrToInt() – функція перетворення типу даних з рядка до цілочислового типу;

```
procedure TForm1.Edit1EditingDone(Sender: TObject);
begin
  Memo1.Lines.Clear;
  a:=StrToInt(Edit1.Text);
  y:=0;
  For i:=0 To a Do
     Begin
       y:=y+i;
     end;
  Memo1.Lines.Add(FloatToStr(y));
end;
```

Процедура натискання на кнопку завершення роботи програми описана у першій роботі.

4. До початку виконання практичної роботи блок-схему алгоритму та текст програмного коду слід затвердити у викладача.
5. Здійснити розв’язання поставленої задачі за допомогою середовища програмування «Lazarus». Після одержання результату узгодити його з викладачем. Виписати отримані результати роботи програми з екрану. Текст програми вивести на принтер. По закінченні роботи проект програми повинен зберігатися на диску ЕОМ у окремому каталозі. Рекомендоване ім’я проекту – LR1_ix де і – порядковий номер прізвища студента по журналу, а **x** – індекс групи.
6. За результатами практичної роботи підготувати звіт, до якого обов’язково входить:

– Назва (тема) практичної роботи.
– Мета роботи.
– Індивідуальне завдання.
– Розробка алгоритму розв’язання задачі:
  - Вибір констант та змінних (таблиця ідентифікаторів).
  - Вигляд форми програми.
  - Графічна схема алгоритму (блок-схема).
– Текст програмного коду.
– Одержані результати та роздруківка тексту програми.
– Формування архіву програм (наводиться ім’я проекту програми, та адреса його зберігання).
– Висновки щодо досягнення мети роботи при розв’язанні наданого індивідуального завдання.

ПРАКТИЧНА РОБОТА №4
ТЕМА: ВИКОРИСТАННЯ ОПЕРАТОРУ ЦИКЛУ З ПЕРЕДУМОВОЮ

Мета: *засвоєння і закріплення теоретичних знань та отримання практичних навичок з використання оператора циклу з передумовою.*

4.1. Загальні положення

Оператор циклу з передумовою виглядає наступним чином:

While <Умова> Do <Дія>;

При виконанні цього оператора спочатку буде перевірено умову і у разі її виконання буде виконано дію. Якщо умова не виконується з початку то цикл (дія) не буде виконана жодного разу.

Виконання роботи потребує попереднього вивчення таких питань теоретичної частини курсу:

1. Загальна структура програми.
2. Стандартні типи даних.
3. Введення та виведення даних.
4. Функції перетворення типів даних.
5. Оператор присвоєння, оператор циклу з передумовою.
6. Порядок обчислення значень арифметичного виразу.

4.2. Постановка задачі та порядок виконання роботи

Задачею роботи є розробка програми з використанням оператора циклу з передумовою яка визначає функцію tg(x) у заданих користувачем межах.

При підготовці до виконання та під час проведення практичної роботи необхідно дотримуватись наступного плану:

1. Уяснити порядок дії програми. Програма повинна вивести результати обчислення функції tg(x) для заданих користувачем початковому і кінцевому значеннях кута з заданим кроком розрахунку.
2. Розробити алгоритм розв'язання задачі. Він складається з:

– Визначення констант і змінних.

Програма повинна мати такі змінні (Табл. 4).

Таблиця 4 – Опис змінних

Параметр	Ім'я змінної	Тип даних
$\varphi_{град}$ – поточне значення кута, град	Fi	Real
$\varphi_{рад}$ – поточне значення кута, рад	Fir	Real
$\varphi_{град}$ – початкове значення кута, град	Fip	Real
$\varphi_{град}$ – кінцеве значення кута, град	Fik	Real
$\Delta\varphi$ – крок розрахунку, град	dFi	Real
y – результат	y	Real

– Розробка форми вікна програми.

Вікно форми програми повинно містити такі компоненти:

Label – надписи з інформацією про номер роботи, про виконавця, підпис компоненту введення кінцевого числа та компоненту з результатом розрахунку (текст записується у полі Caption компоненту).

Edit – компоненти до яких вводяться значення початкового і кінцевого значення кута та кроку розрахунку (у полі Text компонентів слід вказати початкові значення: –10; 10; 1).

Button – кнопка завершення роботи програми (текст, який відображається на ній, записується у полі Caption компоненту).
Memo – до цього компоненту форми буде виводитись результат виконання розрахунку (для очищення компоненту від надпису сліз звернутись до властивості Lines та викликати редактор рядків у якому очистити вміст), (для властивості ScrollBars слід встановити значення ssAutoBoth для появи полоси прокручування в разі виходу тексту за межі компоненту).

Приклад розташування компонентів на формі показано на рис. 7.

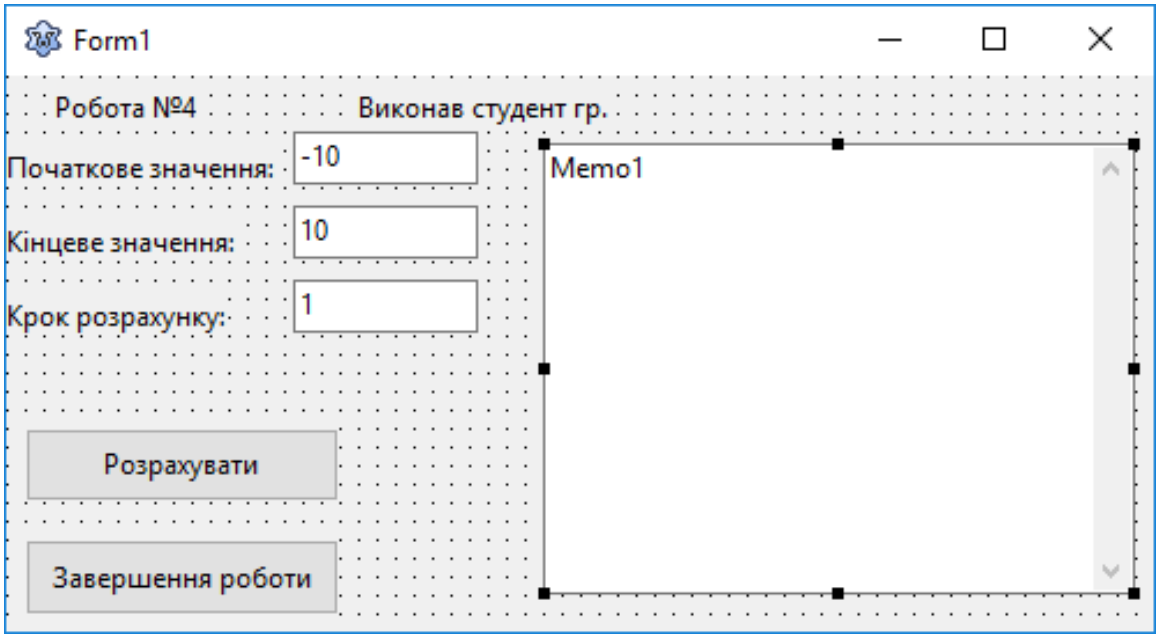


Рисунок 7 – Приклад вигляду форми вікна програми

– Розробка графічної блок-схеми алгоритму.

Алгоритм має розгалужену структуру (рис. 8).

3. Розробити текст програми. При запису тексту додержуватись розробленого алгоритму.

При складанні програмного коду, з початку, слід вказати у блоці var імена змінних з табл. 4 та вказати їх тип даних:

```
var
  Form1: TForm1;
  Fi, Fir, Fip, Fik, dFi, y: Real;
```

Після опису змінних слід створити процедури які будуть виконуватись при натисканні на кнопку з надписом «Розрахувати» (для цього слід двічі клікнути на ній, після чого до тексту програмного коду буде автоматично додано опис нової процедури) та при натисканні на кнопку завершення роботи програми (див. попередню роботу).

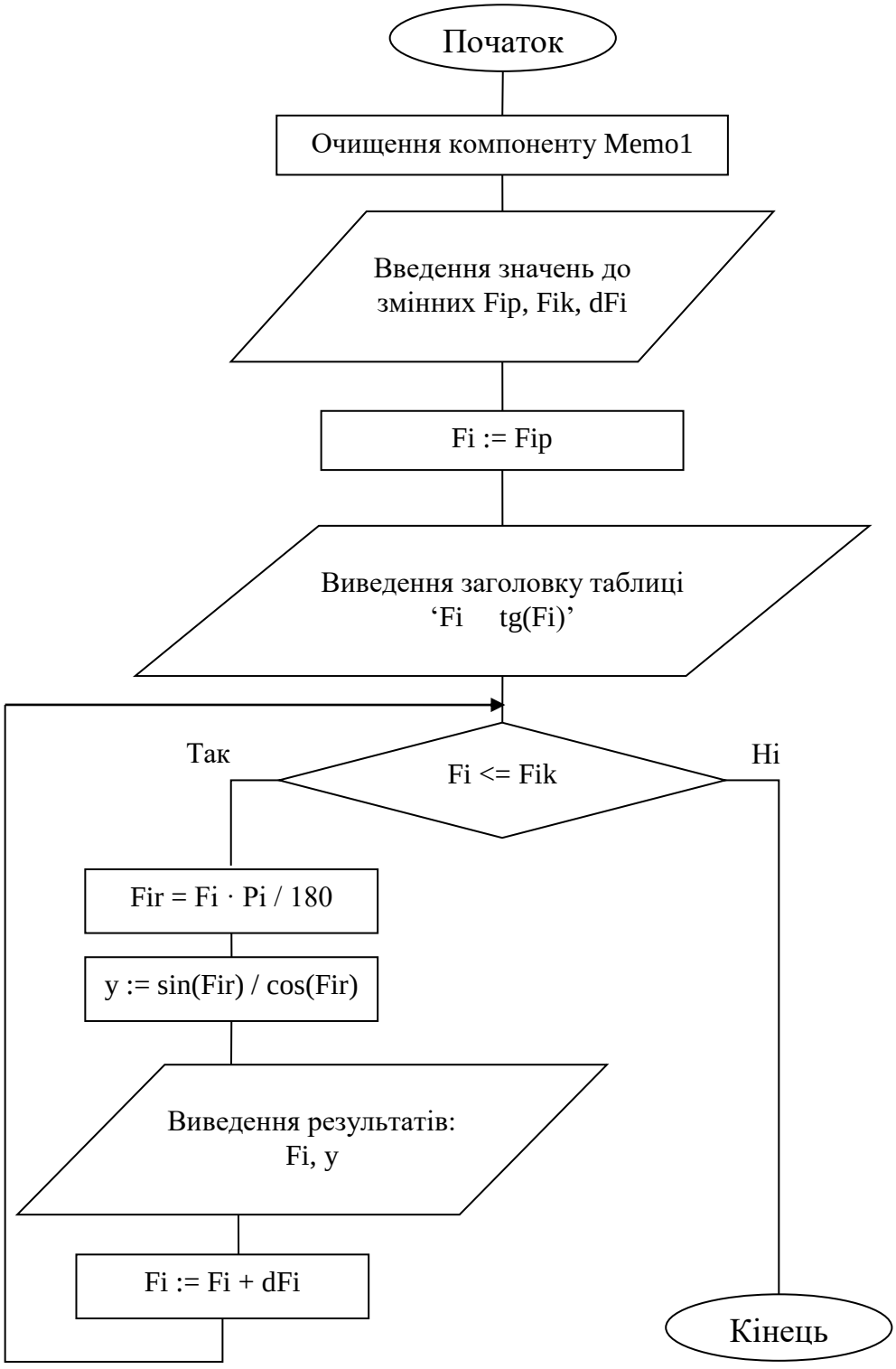


Рисунок 8 – Блок-схема алгоритму дії при натисканні на кнопку з надписом «Розрахувати»

До створених процедур слід додати програмний код який відповідає діям вказаним у блок-схемі. У програмному коді використовуються такі оператори та функції:
:= – оператор привласнення;
While <Умова> Do <Дія>; – оператор циклу з передумовою;
StrToCur() – функція перетворення типу даних з рядка до відповідного числового типу;
FloatToStrf(k, ffFixed, a, b) – функція перетворення типу даних з числового формату до рядка з форматуванням (k – ім'я змінної; a – загальна кількість позицій під число; b – кількість позицій під частину числа після крапки).

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Memo1.Lines.Clear;
  Fip:=StrToCurr(Edit1.Text);
  Fik:=StrToCurr(Edit2.Text);
  dFi:=StrToCurr(Edit3.Text);
  Fi:=Fip;
  Memo1.Lines.Add('Fi      tg(Fi)');
  While Fi<=Fik Do
       Begin
         Fir:=Fi*Pi/180;
         y:=sin(Fir)/cos(Fir);
         Memo1.Lines.Add(FloatToStrf(Fi,ffFixed,6,1)+'
'+FloatToStrf(y,ffFixed,12,7));
         Fi:=Fi+dFi;
       end;
end;
```

Процедура натискання на кнопку завершення роботи програми описана у першій роботі.
4. До початку виконання практичної роботи блок-схему алгоритму та текст програмного коду слід затвердити у викладача.
5. Здійснити розв'язання поставленої задачі за допомогою середовища програмування «Lazarus». Після одержання результату узгодити його з викладачем. Виписати отримані результати роботи програми з екрану. Текст програми вивести на принтер. По закінченні роботи проект програми повинен зберігатися на диску ЕОМ у окремому ка-

талозі. Рекомендоване ім’я проекту – LR1_ix де і – порядковий номер прізвища студента по журналу, а **x** – індекс групи.
6. За результатами практичної роботи підготувати звіт, до якого обов’язково входить:
– Назва (тема) практичної роботи.
– Мета роботи.
– Індивідуальне завдання.
– Розробка алгоритму розв’язання задачі:
  - Вибір констант та змінних (таблиця ідентифікаторів).
  - Вигляд форми програми.
  - Графічна схема алгоритму (блок-схема).
– Текст програмного коду.
– Одержані результати та роздруківка тексту програми.
– Формування архіву програм (наводиться ім’я проекту програми, та адреса його зберігання).
– Висновки щодо досягнення мети роботи при розв’язанні наданого індивідуального завдання.

ПРАКТИЧНА РОБОТА №5
ТЕМА: ВИКОРИСТАННЯ ОПЕРАТОРУ ЦИКЛУ З ПІСЛЯУМОВОЮ

Мета: *засвоєння і закріплення теоретичних знань та отримання практичних навичок з використання оператора циклу з післяумовою*.

5.1. Загальні положення

Оператор циклу з післяумовою виглядає наступним чином:

```
Repeat
    <Дія1>;
    <Дія2>;
    …
    <Дія_n>;
Until <Умова>;
```

Цикл, обмежений ключовими словами Repeat Until буде повторюватись до тих пір доки не буде виконуватись умова. При використ-

танні цього оператору тіло циклу обов'язково буде виконано хоча-би один раз.

Виконання роботи потребує попереднього вивчення таких питань теоретичної частини курсу:

1. Загальна структура програми.
2. Стандартні типи даних.
3. Введення та виведення даних.
4. Функції перетворення типів даних.
5. Оператор присвоєння, оператор циклу з передумовою.
6. Порядок обчислення значень арифметичного виразу.

5.2. Постановка задачі та порядок виконання роботи

В задачу практичної роботи входить розробка програми з використанням оператора циклу з передумовою для визначення функції $y=\sin(x)^n$ у заданих користувачем межах та заданим кроком.

При підготовці до виконання та під час проведення практичної роботи необхідно дотримуватись наступного плану:

1. Уяснити порядок дії програми. Програма повинна вивести результати обчислення функції $y=\sin(x)^n$ для заданих користувачем початковому і кінцевому значеннях кута з заданим кроком розрахунку.
2. Розробити алгоритм розв'язання задачі. Він складається з:
   – Визначення констант і змінних.

   Програма повинна мати такі змінні (Табл. 5).
   – Розробка форми вікна програми.

   Вікно форми програми повинно містити такі компоненти:

Label – надписи з інформацією про номер роботи, про виконавця, підпис компоненту введення кінцевого числа та компоненту з результатом розрахунку (текст записується у полі Caption).

Edit – компоненти до яких вводяться значення початкового і кінцевого значення кута та кроку розрахунку (у полі Text компонентів слід вказати початкові значення: –10; 10; 1).

Button – кнопка завершення роботи програми (текст, який відображається на ній, записується у полі Caption компоненту).

Memo – до цього компоненту форми буде виводитись результат виконання розрахунку (для очищення компоненту від надпису сліз звернутись до властивості Lines та викликати редактор рядків у якому очистити вміст), (для властивості ScrollBars слід встановити значення ssAutoBoth для автоматичної появи полоси прокручування).

Таблиця 5 – Опис змінних

Параметр	Ім’я змінної	Тип даних
$\varphi_{град}$ – поточне значення кута, град	Fi	Real
$\varphi_{рад}$ – поточне значення кута, рад	Fir	Real
$\varphi_{град}$ – початкове значення кута, град	Fip	Real
$\varphi_{град}$ – кінцеве значення кута, град	Fik	Real
$\Delta\varphi$ – крок розрахунку, град	dFi	Real
n – показник ступеня	n	Real
y – результат	y	Real

Приклад розташування компонентів на формі показано на рис. 9.

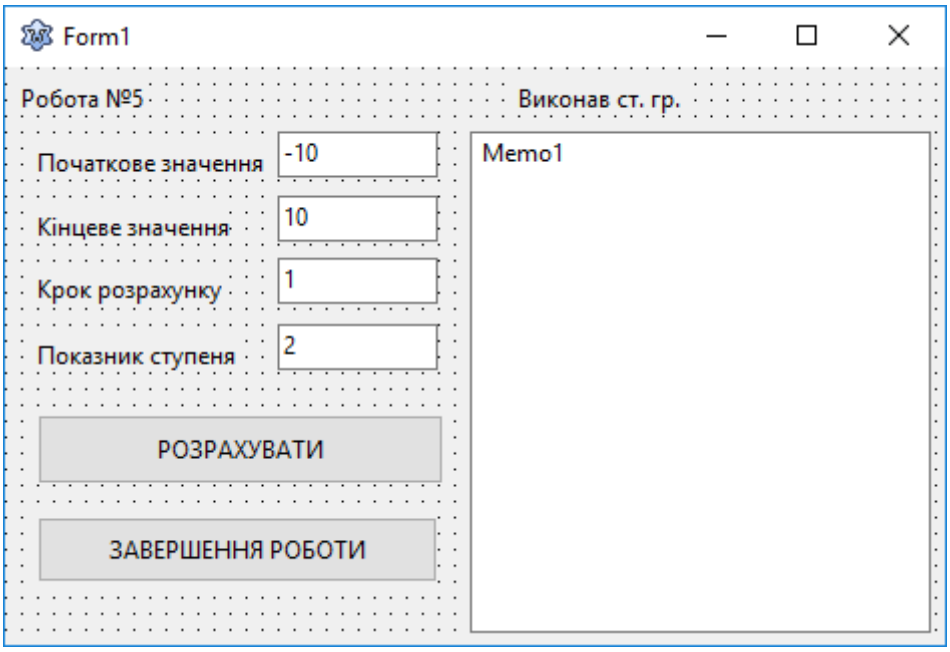


Рисунок 9 – Приклад вигляду форми вікна програми

– Розробка графічної блок-схеми алгоритму.

Алгоритм має розгалужену структуру (рис. 10).

3. Розробити текст програми. При запису тексту додержуватись розробленого алгоритму.

При складанні програмного коду, з початку, слід вказати у блоці var імена змінних з табл. 5 та вказати їх тип даних:

```
var
  Form1: TForm1;
  Fi, Fip, Fik, dFi, n, y: Real;
```

Після опису змінних слід створити процедури які будуть виконуватись при натисканні на кнопку з надписом «Розрахувати» (для цього слід двічі клікнути на ній, після чого до тексту програмного коду буде автоматично додано опис нової процедури) та при натисканні на кнопку завершення роботи програми (див. попередню роботу).

До створених процедур слід додати програмний код який відповідає діям вказаним у блок-схемі. У програмному коді використовуються такі оператори та функції:

:= – оператор привласнення;

Repeat <Дія1>; <Дія2>; …; <Дія_n> Until <Умова>; – оператор циклу з післяумовою;

StrToCur() – функція перетворення типу даних з рядка до відповідного числового типу;

FloatToStrf(k, ffFixed, a, b) – функція перетворення типу даних з числового формату до рядка з форматуванням (k – ім’я змінної; a – загальна кількість позицій під число; b – кількість позицій під частину числа після крапки).

Оскільки функції возведення у довільний ступінь в стандарті мови нема, то для використання такої функції її слід описати окремо:

```
Function Pow(x, y:real):real;
Begin
  If x=0 Then pow:=0 Else pow := Exp(abs(y)*Ln(abs(x)));
  If y=0 Then pow:=1;
  If x<0 Then If (Round(y) mod 2 <> 0) Then pow:=-pow;
End;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Memo1.Lines.Clear;
  Fip:=StrToCurr(Edit1.Text);
  Fik:=StrToCurr(Edit2.Text);
  dFi:=StrToCurr(Edit3.Text);
  n:=StrToCurr(Edit4.Text);
  Fi:=Fip;
```

```
  Memo1.Lines.Add('Fi      sin(Fi)^n');
  Repeat
        y:=pow(Fi,n);
        Memo1.Lines.Add(FloatToStrf(Fi,ffFixed,6,1)+'
'+FloatToStrf(y,ffFixed,12,7));
        Fi:=Fi+dFi;
  Until Fi>Fik;
end;
```

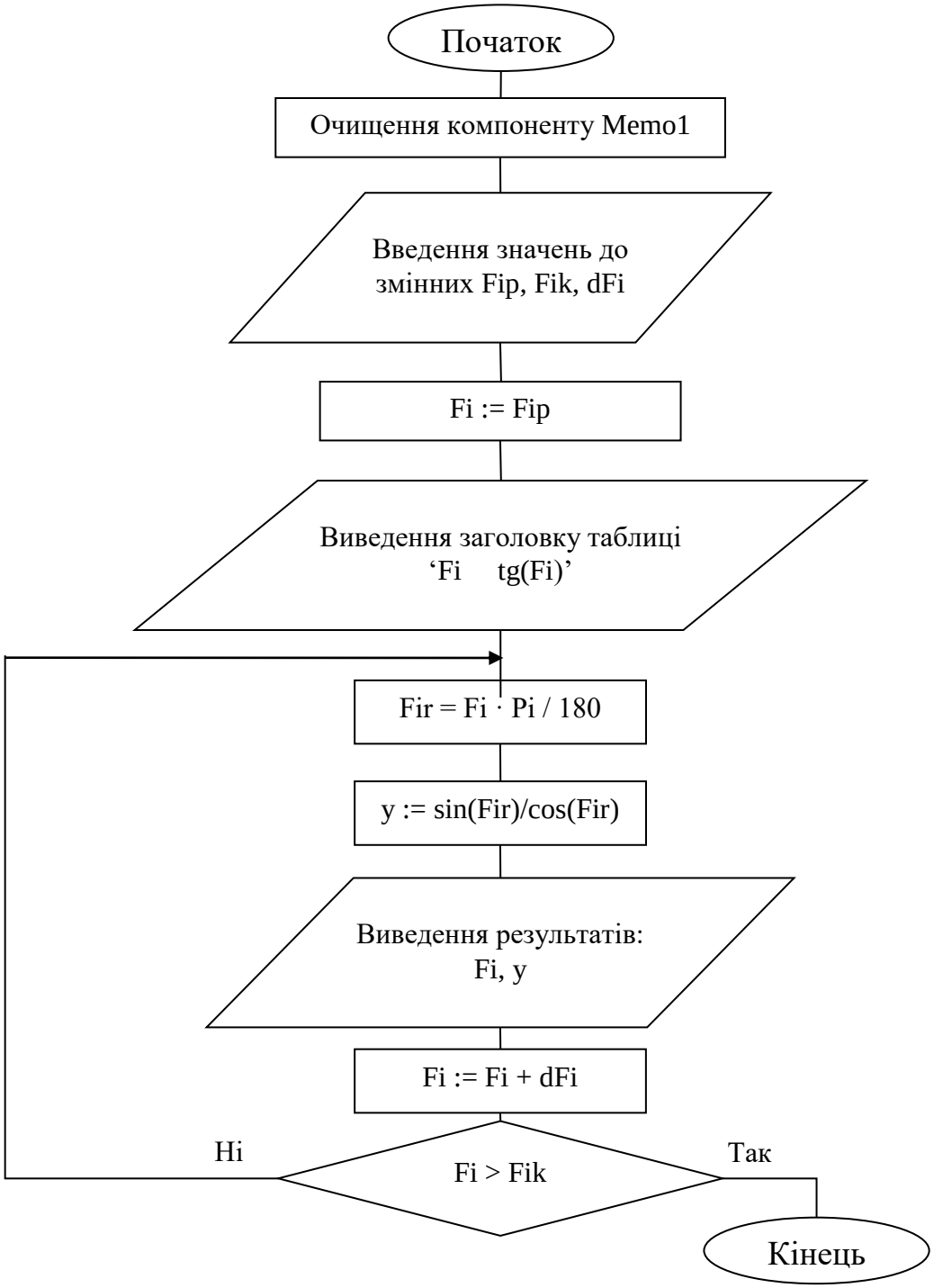


Рисунок 10 – Блок-схема алгоритму дії при натисканні на кнопку з надписом «Розрахувати»

Процедура натискання на кнопку завершення роботи програми описана у першій роботі.

4. До початку виконання практичної роботи блок-схему алгоритму та текст програмного коду слід затвердити у викладача.

5. Здійснити розв'язання поставленої задачі за допомогою середовища програмування «Lazarus». Після одержання результату узгодити його з викладачем. Виписати отримані результати роботи програми з екрану. Текст програми вивести на принтер. По закінченні роботи проект програми повинен зберігатися на диску ЕОМ у окремому каталозі. Рекомендоване ім'я проекту – LR1_iix де ii – порядковий номер прізвища студента по журналу, а **x** – індекс групи.

6. За результатами практичної роботи підготувати звіт, до якого обов'язково входить:

- Назва (тема) практичної роботи.
- Мета роботи.
- Індивідуальне завдання.
- Розробка алгоритму розв'язання задачі:
  - Вибір констант та змінних (таблиця ідентифікаторів).
  - Вигляд форми програми.
  - Графічна схема алгоритму (блок-схема).
- Текст програмного коду.
- Одержані результати та роздруківка тексту програми.
- Формування архіву програм (наводиться ім'я проекту програми, та адреса його зберігання).
- Висновки щодо досягнення мети роботи при розв'язанні наданого індивідуального завдання.

ПРАКТИЧНА РОБОТА №6
ТЕМА: ВИКОРИСТАННЯ КОМПОНЕНТУ «ГРАФІК»

Мета: *засвоєння і закріплення теоретичних знань та отримання практичних навичок з використання компонентів візуалізації даних.*

6.1. Загальні положення

Компонент TChart з вкладки Chart палітри компонентів (рис. 11) дозволяє візуалізувати дані розрахунків у вигляді графіків.

Рисунок 11 –Палітра компонентів Chart

Після додавання компоненту до форми отримуємо «пустий» графік (рис. 12).

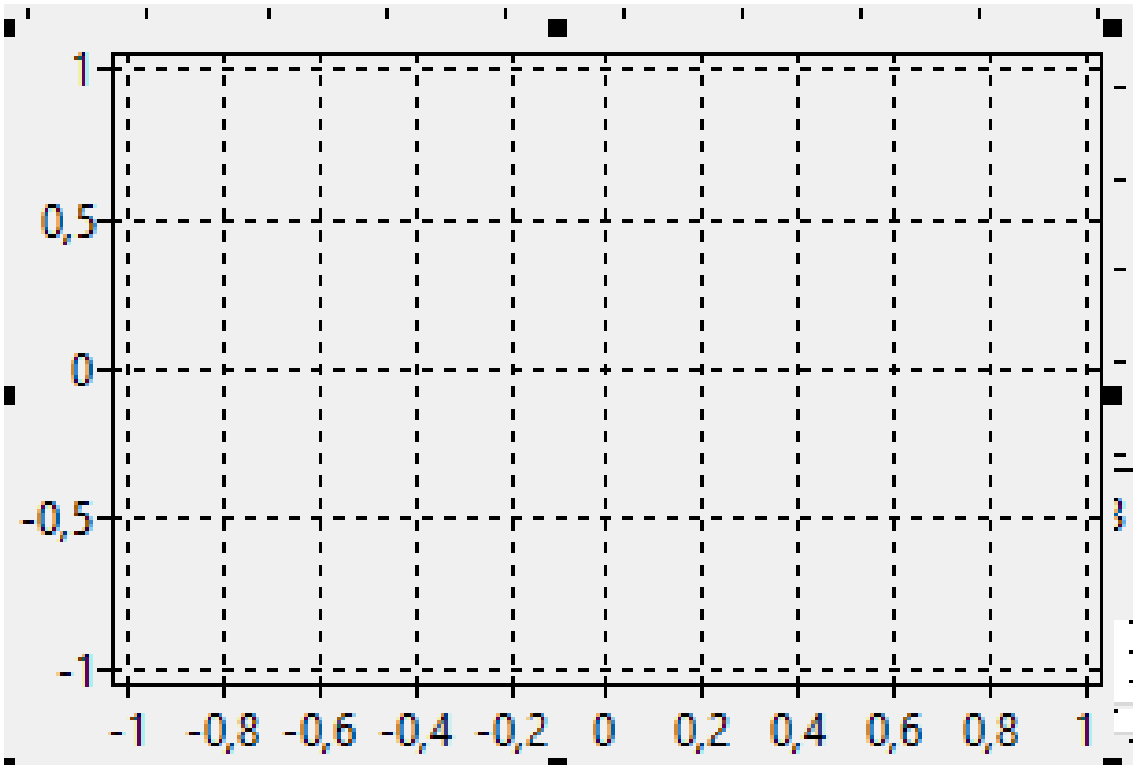


Рисунок 12 – Компонент Chart на формі

Для додавання лінії на графік слід викликати контекстне меню та обрати пункт «Редактор діаграм», - буде відкрито вікно роботи з діаграмою (рис. 13). Тут слід додати необхідну кількість ліній на графік.

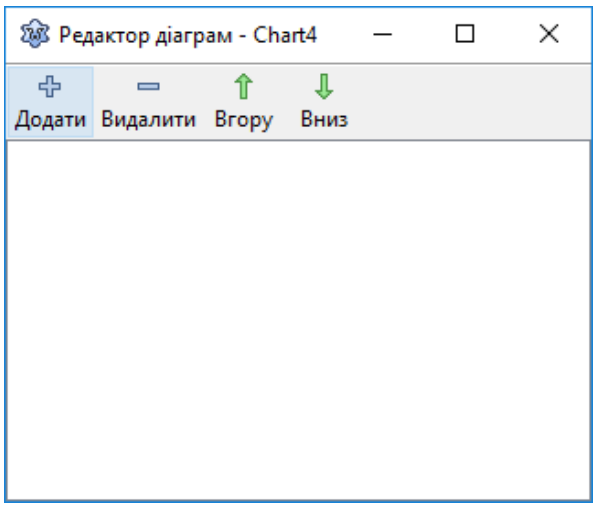


Рисунок 13 – Вікно редактора діаграм компоненту Chart

Налаштування графіків роблять за допомогою інспектора об’єктів. В ньому задають підписи осей діаграми, їх параметри, колір та інші параметри лінії.

Для зручного розташування на формі великої кількості даних зручно використовувати компонент зі сторінками які можна перемикати за їх заголовками. Додавання сторінки до цього компоненту показано на рис. 14.

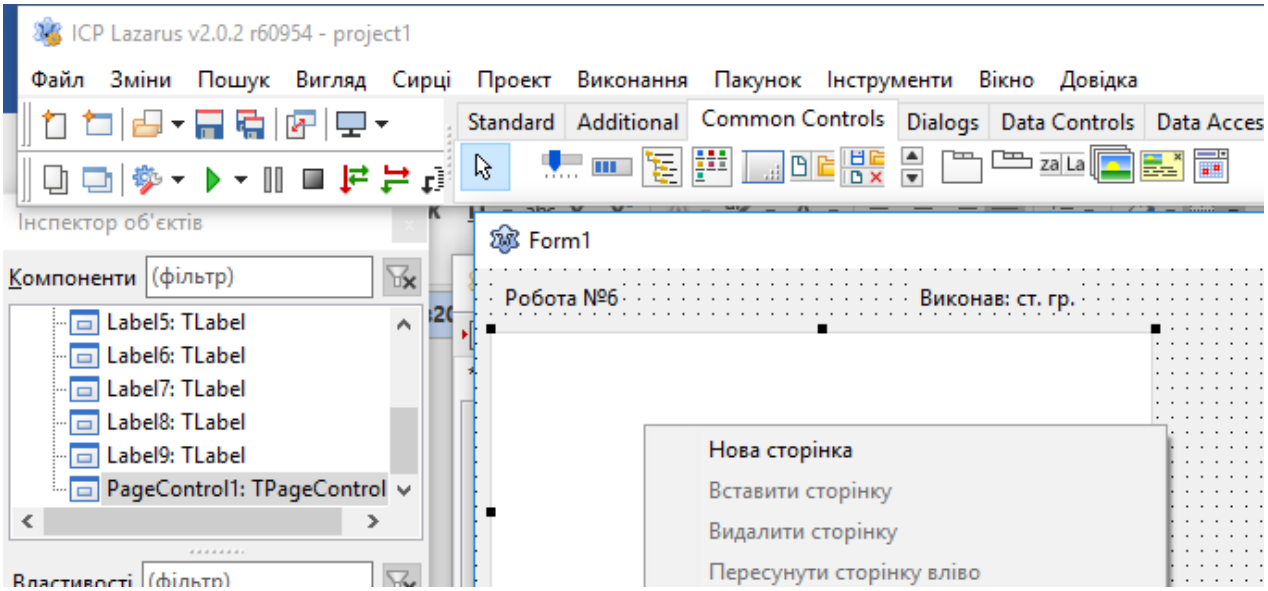


Рисунок 14 – Додавання нової сторінки до компоненту PageControl

Встановлення назви заголовку сторінки відбувається у інспекторі об’єктів через поле Caption (виділеною повинна бути сторінка). На кожну зі сторінок можна додавати власні компоненти.

Виконання роботи потребує попереднього вивчення таких питань теоретичної частини курсу:

1. Загальна структура програми.
2. Стандартні типи даних.
3. Введення та виведення даних.
4. Функції перетворення типів даних.
5. Оператор присвоєння, оператор циклу з передумовою.
6. Порядок обчислення значень арифметичного виразу.
7. Робота з компонентами.

6.2. Постановка задачі та порядок виконання роботи

В задачу практичної роботи входить розробка програми з використанням оператора циклу для визначення функції зміни об'єму циліндру двигуна за один оборот колінчастого валу (від 0º до 360º) з заданим кроком розрахунку та з візуалізацією результату розрахунку.

При підготовці до виконання та під час проведення практичної роботи необхідно дотримуватись наступного плану:

1. Уяснити порядок дії програми. Програма повинна вивести результати обчислення функції поточного об'єму циліндру поршневого двигуна внутрішнього згоряння для заданих користувачем параметрах циліндру (D – діаметр циліндру; S – хід поршня; ε – ступінь стиску), початковому і кінцевому значеннях кута з заданим кроком розрахунку. Залежності для визначення об'єму циліндру:

Робочий об'єм циліндра, м^3:

$$V_h = (\pi \cdot D^2 / 4) \cdot S. \tag{1}$$

Об'єм камери стиску, м^3:

$$V_c = V_h / (\varepsilon - 1). \tag{2}$$

Поточний об'єм циліндра у залежності від кута повороту колінчастого валу, м^3:

$$V_\varphi = V_c + V_h \cdot \sigma / 2, \tag{3}$$

де $\sigma = S_\varphi / R$ – відносне переміщення поршня, знаходять за формулою в залежності від кута повороту колінчастого валу φ і співвідношення $\lambda_{кр} = R/L$

$$\sigma = \frac{S_\varphi}{R} = (1 - \cos(\varphi)) + \frac{\lambda_{кр}}{4}(1 - \cos(2 \cdot \varphi)) \tag{4}$$

(тут $R = S/2$ – радіус кривошипа, L – довжина шатуна).

2. Розробити алгоритм розв’язання задачі. Він складається з:
   – Визначення констант і змінних.

Програма повинна мати такі змінні (Табл. 6).

Таблиця 6 – Опис змінних

Параметр	Ім’я змінної	Тип даних
D – діаметр циліндру	D	
S – хід поршня	S	
ε – ступінь стиску	E	
L – довжина шатуна	L	
V_h – робочий об’єм циліндру	VH	
V_c – об’єм камери стиску	VC	
$\varphi_{град}$ – поточне значення кута, град	Fi	Real
$\varphi_{рад}$ – поточне значення кута, рад	Fir	Real
$\varphi_{град}$ – початкове значення кута, град	Fip	Real
$\varphi_{град}$ – кінцеве значення кута, град	Fik	Real
Δφ – крок розрахунку, град	dFi	Real
i – поточний крок розрахунку	i	Integer
V – результат	V	Array of Real

   – Розробка форми вікна програми.

Вікно форми програми повинно містити такі компоненти:

Label – надписи з інформацією про номер роботи, про виконавця, підпис компоненту введення кінцевого числа та компоненту з результатом розрахунку (текст записується у полі Caption).

Edit – компоненти до яких вводяться значення початкового і кінцевого значення кута та кроку розрахунку (у полі Text компонентів слід вказати початкові значення: –10; 10; 1).

Button – кнопка завершення роботи програми (текст, який відображається на ній, записується у полі Caption компоненту).

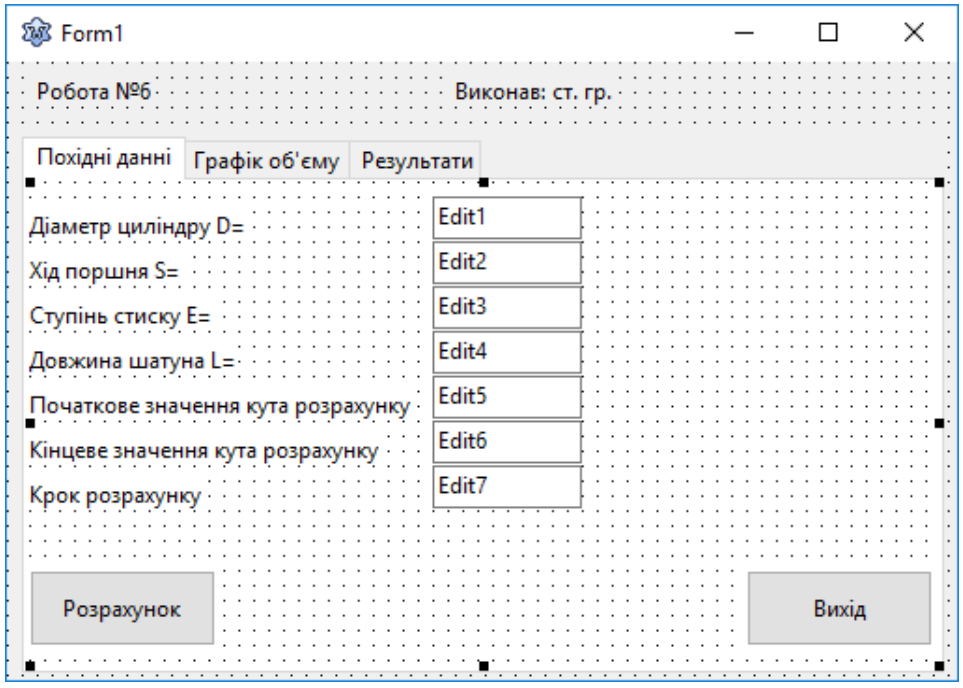
Memo – до цього компоненту форми буде виводитись результат виконання розрахунку (налаштування компоненту описані раніш). Компонент слід розташувати на сторінці «Результати».

Chart – у цьому компоненті буде відображено графік функції що розраховуємо (для очищення компоненту від попередніх даних перед

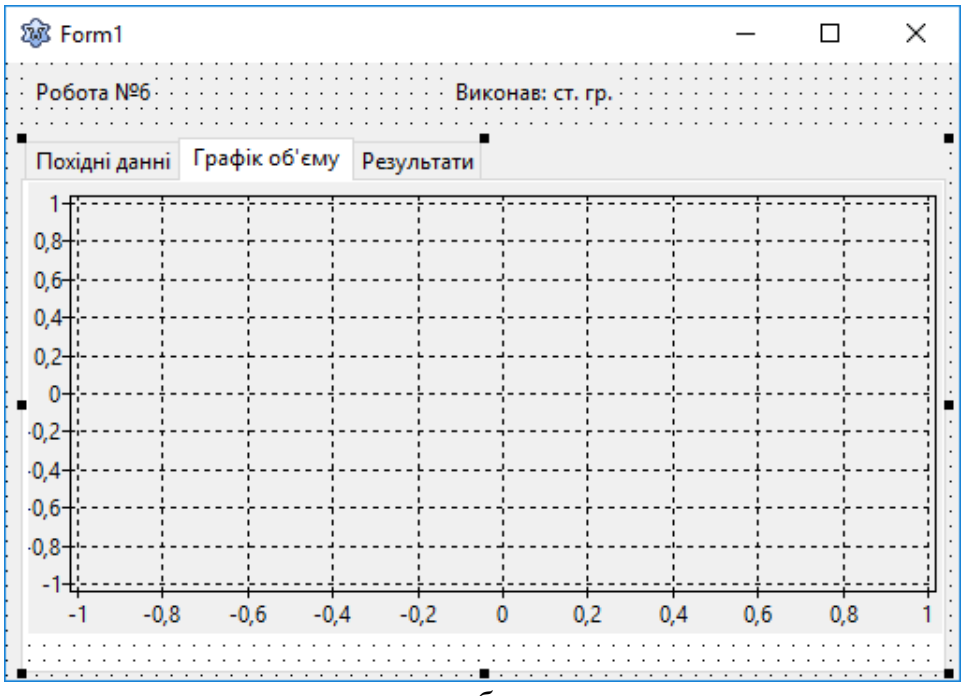
використанням компоненту слід прописувати властивість: Chart1LineSeries1.Clear;). Компонент слід розташувати на сторінці «графік об’єму».

PageControl – для формування трьох сторінок: похідні данні; графік об’єму; результати.

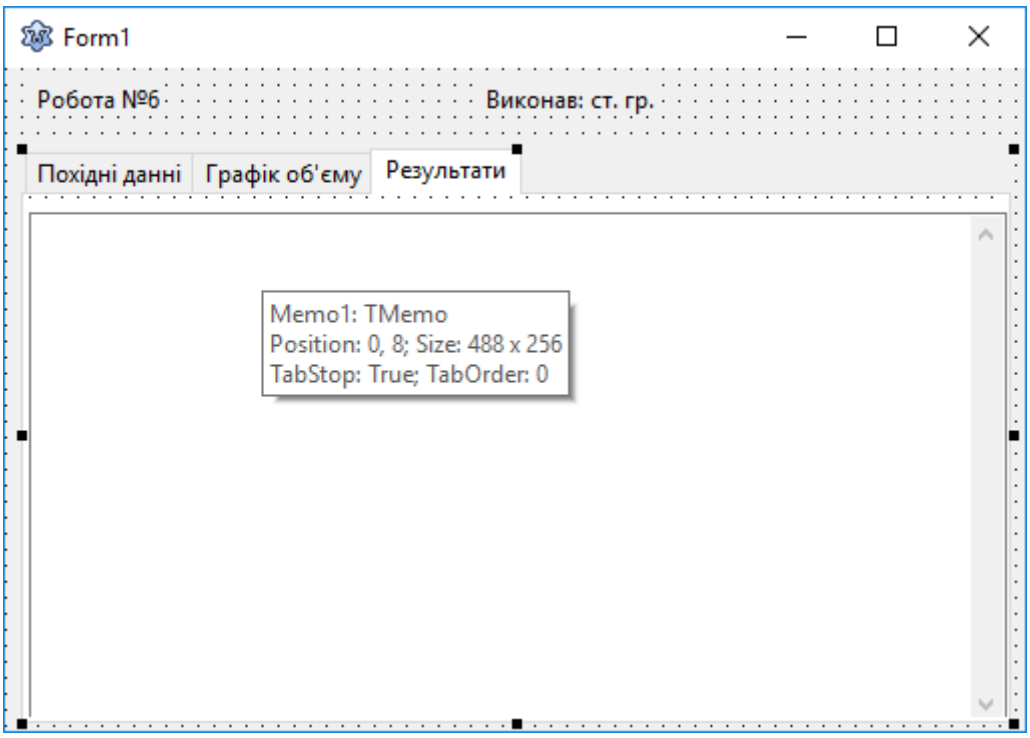
Приклад розташування компонентів на формі показано на рис. 14.



а



б



в

Рисунок 14 – Приклад вигляду форми вікна програми зі сторінкою похідних даних(*а*); зі сторінкою графіка об’єму(*б*); зі сторінкою результатів розрахунків у текстовому вигляді (*в*)

– Розробка графічної блок-схеми алгоритму.

Алгоритм має розгалужену структуру (рис. 10).

3. Розробити текст програми. При запису тексту додержуватись розробленого алгоритму.

При складанні програмного коду, з початку, слід вказати у блоці var імена змінних з табл. 5 та вказати їх тип даних:

```
var
  Form1: TForm1;
  D, S, E, Lam, Fi, Fir, Fip, Fik, dFi, VH, VC: Real;
  I: Integer;
  V: Array [0..100] of Real;
```

Після опису змінних слід створити процедури які будуть виконуватись при натисканні на кнопку з надписом «Розрахувати» (для цього слід двічі клікнути на ній, після чого до тексту програмного коду буде автоматично додано опис нової процедури) та при натисканні на кнопку завершення роботи програми (див. попередню роботу).

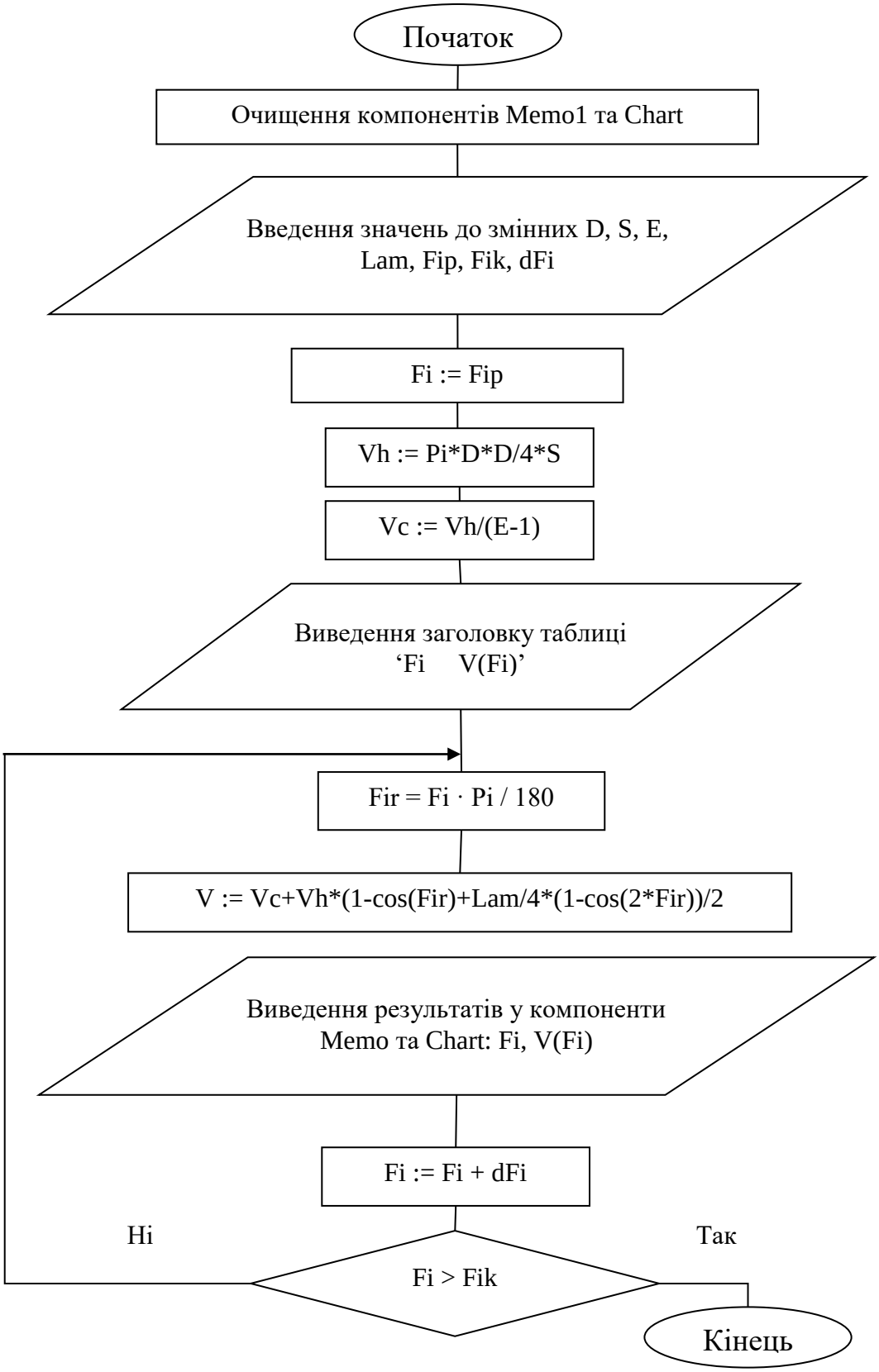


Рисунок 15 – Блок-схема алгоритму дії при натисканні на кнопку з надписом «Розрахувати»

До створених процедур слід додати програмний код який відповідає діям вказаним у блок-схемі. У програмному коді використовуються такі оператори та функції:
:= – оператор привласнення;
Repeat <Дія1>; <Дія2>; …; <Дія_n>Until <Умова>; – оператор циклу з післяумовою;
StrToCur() – функція перетворення типу даних з рядка до відповідного числового типу;
FloatToStrf(k, ffFixed, a, b) – функція перетворення типу даних з числового формату до рядка з форматуванням (k – ім’я змінної; a – загальна кількість позицій під число; b – кількість позицій під частину числа після крапки);
Для автоматичного відкриття сторінки з графіком використовуємо запис: PageControl1.ActivePage:=TabSheet2;.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Memo1.Lines.Clear;
  PageControl1.ActivePage:=TabSheet2;
  D:=StrToFloat(Edit1.Text);
  S:=StrToFloat(Edit2.Text);
  E:=StrToFloat(Edit3.Text);
  Lam:=StrToFloat(Edit4.Text);
  Fip:=StrToFloat(Edit5.Text);
  Fik:=StrToFloat(Edit6.Text);
  DFi:=StrToFloat(Edit7.Text);
  Fi:=Fip;
  I:=1;
  Vh:=Pi*D*D/4*S;
  Vc:=Vh/(E-1);
  Memo1.Lines.Add('Vh='+FloatToStrf(Vh,ffFixed,8,6)+'
Vc='+FloatToStrf(Vc,ffFixed,10,8));
  Memo1.Lines.Add('  Fi      V(Fi)  ');
  Repeat
        Fir:=Fi*Pi/180;
        V[i]:=Vc+Vh*(1-cos(Fir)+Lam/4*(1-cos(2*Fir)))/2;
        Memo1.Lines.Add(FloatToStrf(Fi,ffFixed,6,1)+'
'+FloatToStrf(V[i],ffFixed,12,7));
        Chart1LineSeries1.AddXY(Fi,V[i],floattostr(Fi));
```

```
        Fi:=Fi+dFi;
        I:=I+1;
  Until Fi>Fik;
end;
```

Процедура натискання на кнопку завершення роботи програми описана у першій роботі.

4. До початку виконання практичної роботи блок-схему алгоритму та текст програмного коду слід затвердити у викладача.

5. Здійснити розв'язання поставленої задачі за допомогою середовища програмування «Lazarus». Після одержання результату узгодити його з викладачем. Виписати отримані результати роботи програми з екрану. Текст програми вивести на принтер. По закінченні роботи проект програми повинен зберігатися на диску ЕОМ у окремому каталозі. Рекомендоване ім'я проекту – LR1_iix де ii – порядковий номер прізвища студента по журналу, а **x** – індекс групи.

6. За результатами практичної роботи підготувати звіт, до якого обов'язково входить:

- Назва (тема) практичної роботи.
- Мета роботи.
- Індивідуальне завдання.
- Розробка алгоритму розв'язання задачі:
  - Вибір констант та змінних (таблиця ідентифікаторів).
  - Вигляд форми програми.
  - Графічна схема алгоритму (блок-схема).
- Текст програмного коду.
- Одержані результати та роздруківка тексту програми.
- Формування архіву програм (наводиться ім'я проекту програми, та адреса його зберігання).
- Висновки щодо досягнення мети роботи при розв'язанні наданого індивідуального завдання.

ПРАКТИЧНА РОБОТА №7
ТЕМА: РОБОТА З ФАЙЛАМИ

Мета: *засвоєння і закріплення теоретичних знань та отримання практичних навичок роботи з файлами.*

7.1. Загальні положення

Робота з файлами поділяється на зчитування даних з файлу та запис даних до файлу.

Для читання даних використовують оператори:

Read (f, a1, a1, …); Readln (f, a1, a2, …),

тут: f – ім’я файлової змінної, яка пов’язана з файлом звідки відбувається операція читання даних;

a1, a2, … – імена змінних до яких будуть вноситись зчитані дані.

Оператор Readln відрізняється від Read тим що після його виконання вказівник введення переходить до наступного рядка, а не залишається у місті закінчення операції.

Перед використанням цих операторів слід пов’язати файлову змінну з ім’ям файлу та відчинити його для читання. Ці операції виконують за допомогою операторів:

AssignFile (f, ’name.txt’); – пов’язання файової змінної f з файлом name.txt.

Reset (f); – відкриття файлу пов’язаного з f для читання з нього.

Для обрання файлу користувачем слід використовувати стандартний компонент діалогу відкриття файлу OpenDialog.

Для виведення даних використовують оператори:

Write (f, a1, a1, …);

Writeln (f, a1, a1, …);

тут: f – ім’я файлової змінної, яка пов’язана з файлом до якого відбувається операція виведення даних;

a1, a2, … – імена змінних значення яких виводяться.

Оператор Writeln відрізняється від Write тим що після його виконання вказівник введення переходить до наступного рядка, а не залишається у місті закінчення операції.

Перед використанням цих операторів слід пов’язати файлову змінну з ім’ям файлу та відчинити його для запису. Ці операції виконують за допомогою операторів:

AssignFile (f, ’name.txt’); – пов’язання файової змінної f з файлом name.txt.

Rewrite (f); – відкриття файлу пов’язаного з f для запису до нього.

Для обрання файлу користувачем слід використовувати стандартний компонент діалогу запису файлу SaveDialog.

Виконання роботи потребує попереднього вивчення таких питань теоретичної частини курсу:

1. Загальна структура програми.

2. Стандартні типи даних.
3. Введення та виведення даних.
4. Функції перетворення типів даних.
5. Оператор присвоєння, оператор повторення.
6. Порядок обчислення значень арифметичного виразу.

7.2. Постановка задачі та порядок виконання роботи

В задачу практичної роботи входить розробка програми з використанням зчитування даних з файлу та запису даних до файлу. Задачею є визначення коефіцієнтів *a* та *b* лінійної залежності теплоємності речовини ($\mu C_{\text{розр}}$) від температури за табличними даними.

$$\mu C_{\text{розр}} = a + b \cdot t_{\text{розр}}, \tag{1}$$

$$b = \left(\sum_{i=1}^{k} \left(t_i \cdot \mu C_i \right) - \sum_{i=1}^{k} t_i \cdot \sum_{i=1}^{k} \mu C_i / k \right) \Bigg/ \left(\sum_{i=1}^{k} t_i^{\,2} - \left(\sum_{i=1}^{k} t_i \right)^2 \Big/ k \right), \tag{2}$$

$$a = \left(\sum_{i=1}^{k} \mu C_i - b \cdot \sum_{i=1}^{k} t_i \right) \Big/ k \ , \tag{3}$$

де $\mu C_{\text{розр}}$ – розрахункове значення теплоємності, кДж/(кмоль·К);

$t_{\text{розр}}$ – значення температури, при якій розраховується теплоємність, ˚С;

a, *b* – коефіцієнти, що підлягають визначенню;

t_i – рівень температури і-го експерименту, ˚С;

μC_i – табличне (експериментально визначене) значення теплоємності в і-у експерименті, кДж/(кмоль·К);

k – кількість експериментів.

При підготовці до виконання та під час проведення практичної роботи необхідно дотримуватись наступного плану:

1. Уяснити порядок дії програми. Програма повинна зчитати табличні значення температури та теплоємності речовини, розрахувати коефіцієнти лінійної залежності яка описує функцію зміни теплоємності від температури та вивести результати до файлу.
2. Розробити алгоритм розв’язання задачі. Він складається з:
   – Визначення констант і змінних.

   Програма повинна мати такі змінні (Табл. 7).

Таблиця 7. Опис змінних

Параметр	Ім’я змінної	Тип даних
i – параметр циклу	i	Integer
f – файлова змінна	f	Text
t_i	t	Arrey [1..6] of Real
μC_i		Arrey [1..6] of Real
$\sum t_i$		Real
$\sum \mu C_i$		Real
$\sum \mu C_i \cdot t_i$		Real
$\sum t_i \cdot t_i$		Real
a		Real
b		Real

– Розробка форми вікна програми.

Вікно форми програми повинно містити такі компоненти:

Label – надписи з інформацією про номер роботи, про виконавця, підпис компоненту введення кінцевого числа та компоненту з результатом розрахунку (текст записується у полі Caption).

Edit – компоненти до яких вводяться значення початкового і кінцевого значення кута та кроку розрахунку (у полі Text компонентів слід вказати початкові значення: –10; 10; 1).

Button – кнопка завершення роботи програми (текст, який відображається на ній, записується у полі Caption компоненту).

Memo – до цього компоненту форми буде виводитись результат виконання розрахунку (для очищення компоненту від надпису слід звернутись до властивості Lines та викликати редактор рядків у якому очистити вміст), (для властивості ScrollBars слід встановити значення ssAutoBoth для появи полоси прокручування).

OpenDialog – компонент діалогу відкриття файлу.

Завдання фільтру розширень для переліку файлів задається через властивість Filter:

```
OpenDialog1.Filter := 'Тексти |*.txt|' + 'Усі файли|*.*';
```

Далі у разі отримання імені файлу з діалогу іде його зв’язування з файловою змінною, виконання відкриття файлу для читання з нього та інші дії:

```
if OpenDialog1.Execute and FileExists(OpenDialog1.FileName) then
```

```
  begin
   AssignFile(f, OpenDialog1.FileName);
   Reset (f);
…
  end;
```

SaveDialog – компонент діалогу зберігання файлу.

Завдання фільтру розширень для переліку файлів задається через властивість Filter:

SaveDialog1.Filter := 'Тексты |*.txt|';

Далі у разі отримання імені файлу з діалогу іде його зв'язування з файловою змінною, виконання відкриття файлу для запису до нього та інші дії:

```
if SaveDialog1.Execute then
    begin
     AssignFile(f,SaveDialog1.FileName);
     Rewrite(f);
…
    end;
```

Приклад розташування компонентів на формі показано на рис. 16.

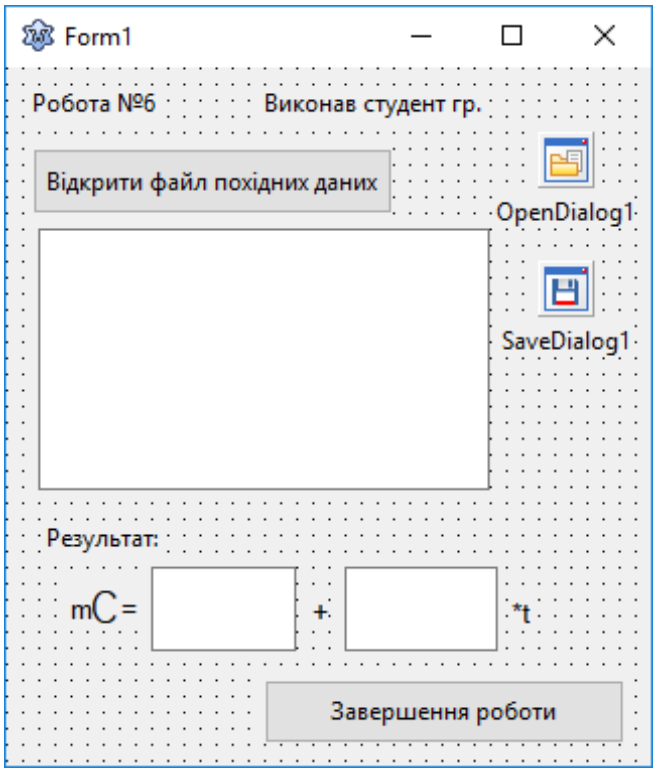
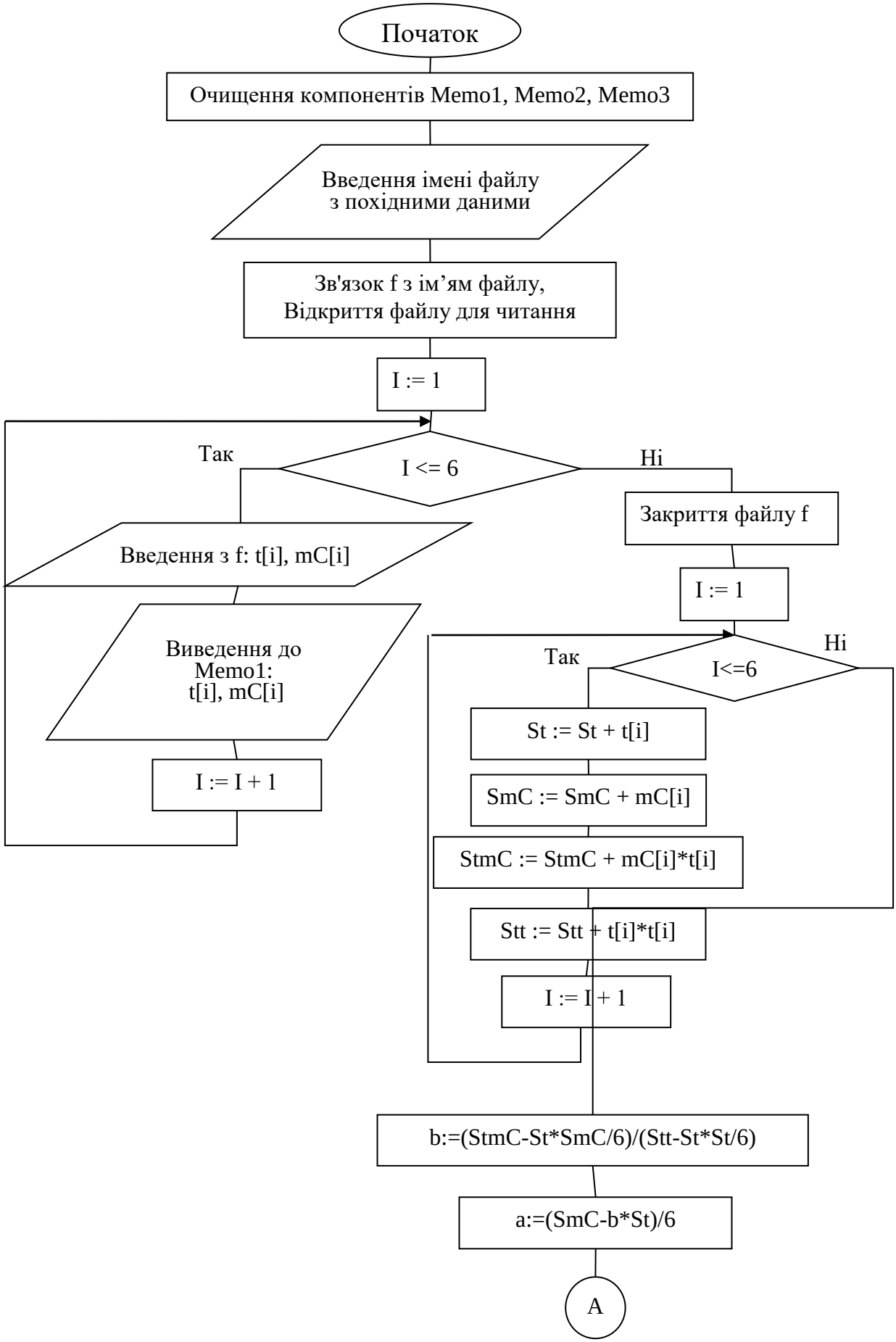


Рисунок 16 – Приклад вигляду форми вікна програми

– Розробка графічної блок-схеми алгоритму.

Алгоритм має розгалужену структуру (рис. 17).



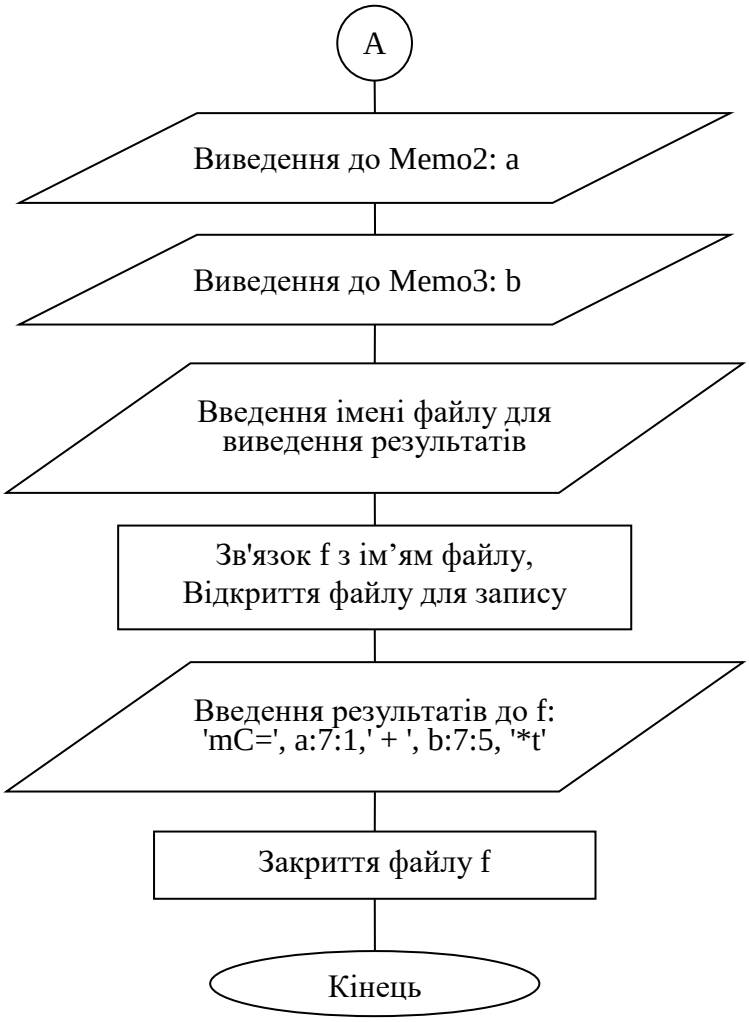


Рисунок 17 – Блок-схема алгоритму дії при натисканні на кнопку з надписом «Розрахувати»

3. Розробити текст програми. При запису тексту додержуватись розробленого алгоритму.

При складанні програмного коду, з початку, слід вказати у блоці var імена змінних з табл. 6 та вказати їх тип даних:

```
var
  Form1: TForm1;
  f: Text;
  i: Integer;
  t, mC: Array[1..6] of Real;
  St, SmC, StmC, Stt, a, b: Real;
```

Після опису змінних слід створити процедури які будуть виконуватись при натисканні на кнопку з надписом «Розрахувати» (для цього слід двічі клікнути на ній, після чого до тексту програмного коду буде автоматично додано опис нової процедури) та при натисканні на кнопку завершення роботи програми (див. попередню роботу).

До створених процедур слід додати програмний код який відповідає діям вказаним у блок-схемі. У програмному коді використовуються такі оператори та функції:

:= – оператор привласнення;

Repeat <Дія1>; <Дія2>; …; <Дія_n> Until <Умова>; – оператор циклу з післяумовою;

StrToCur() – функція перетворення типу даних з рядка до відповідного числового типу;

FloatToStrf(k, ffFixed, a, b) – функція перетворення типу даних з числового формату до рядка з форматуванням (k – ім’я змінної; a – загальна кількість позицій під число; b – кількість позицій під частину числа після крапки).

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.Lines.Clear;
  Memo2.Lines.Clear;
  Memo3.Lines.Clear;
  OpenDialog1.Filter := 'Тексти |*.txt|' + 'Усі файли|*.*';
  if OpenDialog1.Execute and FileExists(OpenDialog1.FileName) then
   begin
    AssignFile(f, OpenDialog1.FileName);
    Reset (f);
    For I:=1 To 6 Do  Begin
       Readln(f, t[i], mC[i]);
       Memo1.Lines.Add(FloatToStrf(t[i],ffFixed,7,2)+'
'+FloatToStrf(mC[i],ffFixed,7,2));
    end;
    CloseFile (f);
    For I:=1 To 6 Do
    Begin
      St:=St+t[i];
      SmC:=SmC+mC[i];
      StmC:=StmC+t[i]*mC[i];
```

```
      Stt:=Stt+t[i]*t[i];
    end;
    b:=(StmC-St*SmC/6)/(Stt-St*St/6);
    a:=(SmC-b*St)/6;
    Memo2.Lines.Add(FloatToStrf(a,ffFixed,7,4));
    Memo3.Lines.Add(FloatToStrf(b,ffFixed,7,5));
    SaveDialog1.Filter := 'Тексти |*.txt|';
    if SaveDialog1.Execute then   begin
     AssignFile(f,SaveDialog1.FileName);
     Rewrite(f);
     Writeln(f, 'mC=', a:7:1,' + ', b:7:5, '*t');
     CloseFile (f);
     end;
   end;
end;
```

Процедура натискання на кнопку завершення роботи програми описана у першій роботі.

4. Створити текстовий файл (без форматування) з даними залежності мольної теплоємності водню при сталому об’ємі від температури (табл. 8). У файлі повинні бути присутні два стовпчика числових значень розділені пробілами.

Таблиця 8. Мольна теплоємність водню в ідеально-газовому стані

t, ˚C	μC_v, кДж/(кмоль·К)
0	20,302
100	20,812
200	20,926
300	20,984
400	21,08
500	21,244

5. Здійснити розв’язання поставленої задачі за допомогою середовища програмування «Lazarus». Після одержання результату узгодити його з викладачем. Виписати отримані результати роботи програми з екрану. Текст програми вивести на принтер. По закінченні роботи проект програми повинен зберігатися на диску ЕОМ у окремому каталозі. Рекомендоване ім’я проекту – LR1_ix де і – порядковий номер прізвища студента по журналу, а **x** – індекс групи.

6. За результатами практичної роботи підготувати звіт, до якого обов'язково входить:

– Назва (тема) практичної роботи.
– Мета роботи.
– Індивідуальне завдання.
– Розробка алгоритму розв'язання задачі:
  - Вибір констант та змінних (таблиця ідентифікаторів).
  - Вигляд форми програми.
  - Графічна схема алгоритму (блок-схема).
– Текст програмного коду.
– Одержані результати та роздруківка тексту програми.
– Формування архіву програм (наводиться ім'я проекту програми, та адреса його зберігання).
– Висновки щодо досягнення мети роботи при розв'язанні наданого індивідуального завдання.

Список літератури

1. Свободное программное обеспечение. FREE PASCAL для студентов и школьников / Ю.Л. Кетков, А.Ю. Кетков. // СПб.: БХВ-Петербург, 2011.
2. Самоучитель по программированию на Free Pascal и Lazarus / Алексеев Е.Р., Чеснокова О.В., Кучер Т.В. // Донецк: ДонНТУ, Технопарк ДонНТУ УНИТЕХ, 2011.
3. Офіційний сайт проекту Lazarus [Електронний ресурс] : [Веб-сайт]. – Електронні дані. Режим доступу: http://www.lazarus-ide.org (дата звернення 05.12.2017) – Назва з екрана.
4. Free Pascal / Lazarus документація [Електронний ресурс] : [Веб-сайт]. – Електронні дані. Режим доступу: http://wiki.freepascal.org (дата звернення 05.12.2017) – Назва з екрана.

ЗМІСТ

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання практичних робіт
«Розробка програм у середовищі «Lazarus»
з дисципліни «Інформаційні технології та програмування в ДВЗ»
для студентів спеціальності: 142 «Енергетичне машинобудування»

Укладачі: ЛІНЬКОВ Олег Юрійович

Відповідальний за випуск проф. Пильов В.О.
До видання роботу рекомендував проф. Шелковой О.М.

Редактор

План 2020 р., поз. ___.
Формат 60x84 1/16.
Гарнітура Times New Roman. Ум. друк. арк. 2,2.

Видавничий центр НТУ «ХПІ». 61002, Харків, вул. Кирпичова, 2.
Свідоцтво про державну реєстрацію ДК №5478 від 21.08.2017 р.

Самостійне електронне видання