Урок №\_\_\_\_\_\_\_\_\_\_\_\_Дата-\_\_\_\_\_\_\_\_\_\_\_\_\_-\_

|  |  |
| --- | --- |
| Тема: | Складання та виконання алгоритмів. Інструктаж з БЖД. Практична робота 12. Складання та виконання алгоритмів з повтореннями та розгалуженнями для опрацювання величин. |
| Мета: | * *освітня:*формувати вміння складати та виконувати алгоритми з повтореннями та розгалуженнями для опрацювання величин в середовищі програмування Lazarus, застосовувати ці вміння для опрацювання величин у цьому середовищі; * *розвивальна*: розвивати аналітичне мислення, пам’ять та увагу; * виховна: виховувати інтерес до інформатики, зосередженість при виконанні завдань, свідоме ставлення до вибору майбутньої професії. |
| Тип уроку: | Застосування вмінь та навичок |
| Обладнання: | Комп'ютери, підручники, інструкції з техніки безпеки, середовище програмування Lazarus, програма для демонстрації на екран чи комп'ютери учнів в мережі. |

Хід уроку

1. *Організаційний момент.*
2. Мотивація вивчення теми.

Сьогодні у нас узагальнюючий урок з теми «Алгоритми роботи з об’єктами та величинами». На цьому уроці ми повторимо теоретичну частину даної теми та застосуємо наші знання на практиці.

1. Узагальнення і систематизація знань.

Давайте пригадаємо, а що ми вже знаємо про алгоритми роботи з об’єктами та величинами?

Робота в парах. Взаємоопитування.

Орієнтовний перелік питань.

1. Поясніть принципи об’єктно-орієнтованого програмування*.* Поясніть, яким чином можуть бути представлені алгоритми з використанням структурних схем.
2. Опишіть етапи створення нового проекту.
3. Які помилки можуть виникнути на кожному з етапів, як їх визначити.
4. Зобразіть циклічні структури алгоритму з використанням блок-схем та коду. Наведіть приклади їх використання.
5. Вкажіть, які алгоритмічні структури відносяться до базових.
6. Представте базові структурні схеми та запис з використанням коду.
7. Поясніть, для чого необхідне середовище програмування.
8. Опишіть структуру типової програми.
9. Поясніть принципову різницю між типами змінних, які можуть бути використані при написанні програм.
10. Опишіть на прикладі результати виконання різних логічних операцій.
11. Опишіть, які оператори умовної передачі управління можуть бути.

IV. Застосування вмінь та навичок.  
Практична робота №12. Інструктаж з БЖД.

1. Виконання комплексу вправ для зняття зорової втоми.

Учитель, враховуючи індивідуальні особливості учнів класу, самостійно визначає час і термін проведення комплексу вправ під час роботи (як правило, через 8-10 хвилин після початку роботи).

1. **Довідка до практичної роботи №12.**

* **алгоритми з повтореннями:**

Оператори циклу дозволяють організувати багатократне повторення однієї і тієї ж послідовності дій. Є три оператори, що дозволяють це зробити:

• "простий" оператор циклу

• умовний оператор циклу

• умовний оператор повторення

Простий оператор циклу застосовується, коли відома кількість повторень циклу. Він записується так:

for лічильник := вираз-1 to вираз-2 do дія

Лічильник - це змінна, яка має бути оголошена перед логічним блоком, в у якому оператор циклу розташований, і її тип повинен належати до одного з типів, що перераховують, звичайно integer. Вираз-1 і вираження-2 можуть бути як константою так і ідентифікатором, так і викликом функції. Дія - один або декілька операторів програми. Якщо це група операторів, то вони мають бути поміщені в логічні дужки begin/end. На початку роботи оператора змінна-лічильник набуває значення виразу-1. Якщо при цьому значення лічильника виявиться менше або дорівнює значенню-2, то виконуються оператори, що входять в дію. Це і є один цикл. Потім змінна-лічильник набуває значення, наступного за поточним, і починається новий цикл, тобто порівняння лічильника і виразу-2, виконання дії, і так далі, до тих пір, поки значення змінної-лічильника не перевищить значення вираження-2. Можлива робота оператора циклу, при якому змінна-лічильник не збільшуватиметься, а зменшуватиметься.

В цьому випадку ключове слово to замінюється на downto:

for лічильник := вираз-1 downto вираз-2 do дія;

Відповідно, вираз-1 має бути більше або дорівнювати виразу-2. Умовний оператор циклу зручно використовувати у тому випадку, коли кількість повторень заздалегідь не відома:

while умова do

тіло циклу ;

Цей цикл виконуватиметься до тих пір, поки істинна умова (логічне вираження, що повертає значення типа Boolean). При цьому якщо цей вираз відразу рівний false, тіло циклу не буде виконано жодного разу.

Потрібно дуже уважно стежити за написанням умови і контролем завершення циклу, оскільки в результаті помилки цикл while повторюватиметься нескінченна кількість разів, що призведе до "зациклення" і "зависання" програми.

Умовний оператор повторення спочатку виконує тіло циклу, а потім вже перевіряє виконання умови:

repeat

тіло циклу

until умова ;

Таким чином, цей варіант циклу гарантує, що тіло циклу буде виконано принаймні один раз. І виконуватиметься до тих пір, поки тіло циклу буде виконано принаймні один раз. І виконуватиметься до тих пір, поки умова не стане істинною (тобто true). Варто відзначити, що це єдиний оператор , в якому тіло циклу не потрібно вкладати в логічні дужки begin/end. Початок і кінець тіла циклу визначаються за ключовими словами repeat і until.

* **алгоритми з розгалуженнями:**

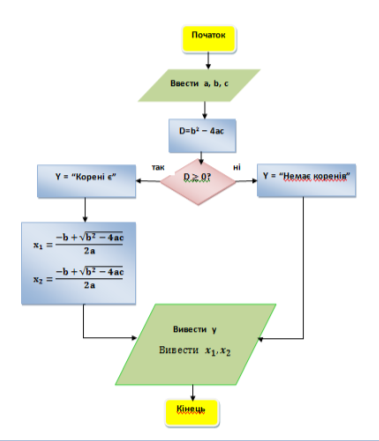
Довідковий матеріал по алгоритмах з розгалуженнями дивись у конспекті уроку №40.

Іноді зручно виводити повідомлення не в написі на формі, а в спеціальне окреме вікно. Якщо таку команду включити до процедури, для виведення результату замість останньої команди в напис або після останньої команди ввести таку: ShowMessage();

1. ***Задача 1.*** Скласти програму для розв’язування квадратного рівняння.

**Вказівки до виконання.**

* Побудуємо блок-схему для розв’язання даної задачі.
* Проаналізуємо разом з учнями дану блок - схему, звертаючи увагу на команду розгалуження та її виконання.
* Повторюємо правила запису умовного оператора.



* Відкриваємо середовище програмування Lazarus.
* Створюємо у власній папці папку **Проекти,** а в ній папку**«Задача 1»,** куди і зберігаємо **Project1,** натиснувшивказівку **Файл/Сохранить всё.**

Виділивши форму, змінимо її властивості у Інспекторі об’єктів на такі:

Caption= Квадратне рівняння,

Color = ‘жовтий’.

За відображення тексту відповідають компоненти Label, Edit(рядок введення), Label(мітка)iButton(кнопка).

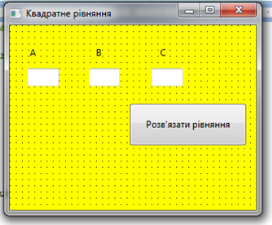
Вибираючи одним клацанням мишки або клавішею Tab потрібний компонент, в Інспекторі oб'єктів ми можемо дістати доступ до його властивостей.

Ці властивості отримують значення в той момент, коли компоненти потрапляють на форму.

За візуальне представлення відповідають для компонентів:

* *Label і Button* - властивість **Caption**,
* *Edit* - властивість  **Text**,
* Розмістимо на формі 3 поля для вставки *Edit*  , 3 мітки для напису *Label* і кнопку *Button* і ще одну мітку для введення результату.
* Змінимо їх властивості в Інспекторі oб'єктів, виділяючи послідовно кожен об'єкт:
* *Edit* - властивість  **Text** очистити,
* *Button* - властивість **Caption**, написати **Розв’язати рівняння.**
* Потім - компонент ***Label*** (*мітка*). У властивості ***Font*** для наочності можна змінити розмір шрифту і колір. Властивість ***Caption*** також очистити.

Отримаємо форму такого вигляду:



Двічі клацнемо по кнопці ***Button1***. Автоматично ми попадаємо у вікно модуля, де система Lazarus створила для нас заготовку обробника події *натискання на кнопку.*

Отже, після вибору кнопки **Button1**(подія **Click**) настає подія **OnClick**, яка полягає у виконанні команд процедури **TForm1.Button1Click**.

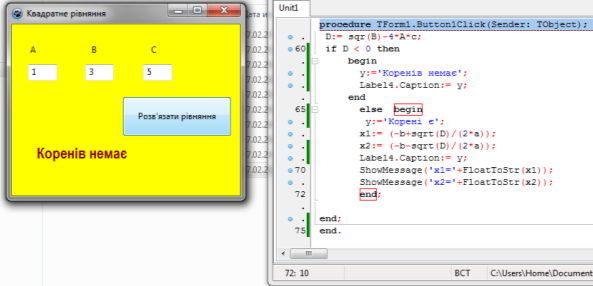
Автоматично курсор зупиняється в тому місці, де необхідно ввести текст процедури**TForm1.Button1Click.**

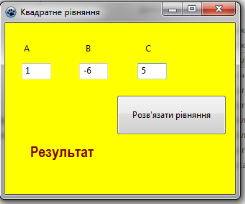
Число, яке вводиться в поле, стає значенням його властивості **Text** і сприймається програмою як текст. Тому перші дві команди процедури є команди присвоювання, які призначені для переведення текстового представлення дійсних чиселу самі дійсні числа (для цього використана функція **StrToFloat ()** і присвоювання цих чисел змінним ***х*** і ***у*** відповідно.

Далі записуються основні команди обчислення результату.

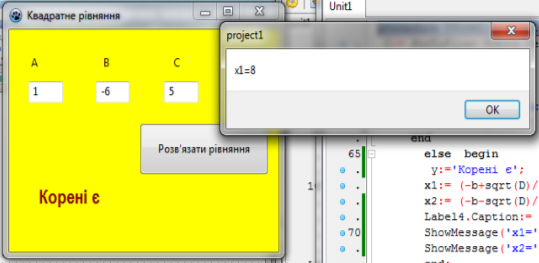
Остання команда переводить результат додавання (значення змінної) з числа в текст, використовуючи функцію **FloatToStr()** і присвоює цей текст значенню властивості **Caption** напису з іменем **Label1** для виведення результату.

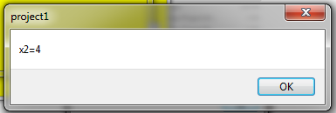
В нашому випадку для виведення результату замість останньої команди в напис введемо таку: ShowMessage(‘х1 = ’+FloatToStr(x1)) для виведення числового результату в окреме вікно.

Отримаємо результат:



або



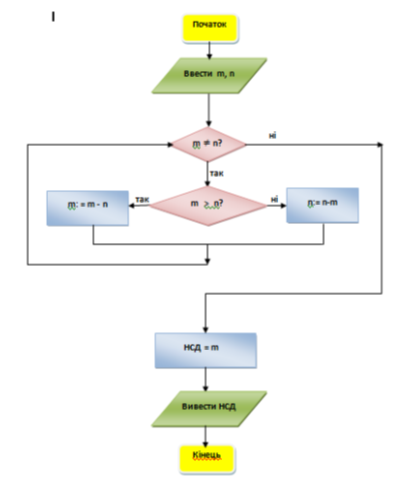


Збережемо проект.

***в).Задача 2.*** Скласти програму пошуку найбільшого спільного дільникадвох даних чисел.

**Вказівки до виконання.**

* Побудуємо блок-схему для розв’язання даної задачі.
* Проаналізуємо разом з учнями дану блок - схему, звертаючи увагу на команду циклу та її виконання.
* Повторюємо правила запису оператора циклу .



* Відкриваємо середовище програмування Lazarus.
* Створюємо у власній папці папку **Проекти,** а в ній папку**«Задача 2»,** куди і зберігаємо **Project2,** натиснувшивказівку **Файл/Сохранить всё.**

Виділивши форму, змінимо її властивості у Інспекторі об’єктів на такі:

Caption= НСД(Алгоритм Евкліда),

Color = ‘оливковий’.

За відображення тексту відповідають компоненти Label, Edit(рядок введення), Label(мітка)iButton(кнопка).

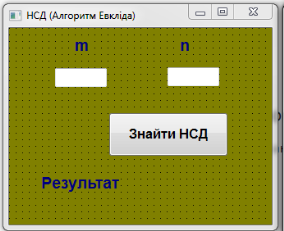
Вибираючи одним клацанням мишки або клавішею Tab потрібний компонент, в Інспекторі oб'єктів ми можемо дістати доступ до його властивостей.

Ці властивості отримують значення в той момент, коли компоненти потрапляють на форму.

За візуальне представлення відповідають для компонентів:

* *Label і Button* - властивість **Caption**,
* *Edit* - властивість  **Text**,
* Розмістимо на формі 3 поля для вставки *Edit*  , 3 мітки для напису *Label* і кнопку *Button* і ще одну мітку для виведення результату.
* Змінимо їх властивості в Інспекторі oб'єктів, виділяючи послідовно кожен об'єкт:
* *Edit* - властивість  **Text** очистити,
* *Button* - властивість **Caption**, написати **Знайти НСД.**
* Потім - компонент ***Label*** (*мітка*). У властивості ***Font*** для наочності можна змінити розмір шрифту і колір. Властивість ***Caption*** також очистити.

Отримаємо форму такого вигляду:

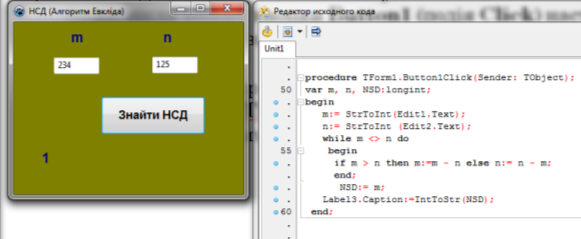


Двічі клацнемо по кнопці ***Button1***. Автоматично ми попадаємо у вікно модуля, де система Lazarus створила для нас заготовку обробника події *натискання на кнопку.*

Отже, після вибору кнопки **Button1**(подія **Click**) настає подія **OnClick**, яка полягає у виконанні команд процедури **TForm1.Button1Click**.

Автоматично курсор зупиняється в тому місці, де необхідно ввести текст процедури**TForm1.Button1Click.**

Отримаємо результат:



Збережемо проект.

**V.Підсумок уроку.**

Рефлексія.

Наприклад, учням пропонується закінчити речення на зразок: "Для мене сьогодні важливим було...", "Сьогодні я дізнався про...", "Мені хотілося в майбутньому дізнатись про..., навчитись …".

* 1. Перевірка, аналіз і оцінка робіт, виконаних під час уроку.

VI.Домашнє завдання

* Опрацювати відповідний параграф підручника, конспект уроку.
* Повторити тему «Алгоритми з повторенням та розгалуженням».
* Скласти блок-схему для розв’язування лінійного рівняння.