МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

Г.Г. ШВАЧИЧ, О.А. ГУЛЯЄВА, Л.М. ПЕТРЕЧУК

ОСНОВИ ПРОГРАМУВАННЯ

Затверджено вченою радою НМетАУ
як навчальний посібник для студентів
спеціальності 151 – автоматизація та
комп'ютерно-інтегровані технології
(бакалаврський рівень).
протокол № 4 від 26.03.18

Дніпро НМетАУ 2018

УДК
Швачич Г.Г., Гуляєва О.А., Петречук Л.М. Основи програмування: Навч. посібник. − Дніпро: НМетАУ, 2018. – 95 с.

Навчальний посібник є керівництвом для студентів, які вивчають об'єктно-орієнтоване програмування, середовище Borland C ++ Builder. Розглянуто основні елементи мов Visual Basic for Applications, C/C++. Викладено принципи візуального і подієвого програмування. Виклад проілюстровано великою кількістю прикладів.

Наведені варіанти завдань для самостійної роботи. У додатку на конкретних прикладах показані основні можливості мови Visual Basic for Applications, а також можливості візуального середовища розробки C++ Builder.

Посібник призначений для студентів спеціальності 151 – автоматизація та комп'ютерно-інтегровані технології (бакалаврський рівень).

Іл. 27. Табл. 4. Бібліогр.: 8 найм.

Друкується за авторською редакцією.

Відповідальний за випуск Г. Г. Швачич, д-р техн. наук, проф.

Рецензенти : Б.І.Мороз, д-р техн. наук, проф. (Академія таможенної служби України)

О.Г.Холод, канд. техн. наук, проф. (Дніпровський університет ім. Альфреда Нобеля)

Зміст

ВСТУП

Наше сьогоднішнє життя нерозривно пов'язане з різними інформаційними процесами, хід яких забезпечується IT-технологіями. Знання цих технологій і основ програмування потрібні в багатьох професіях, починаючи від програмістів, системних адміністраторів і закінчуючи аналітиками, менеджерами, маркетологами.

Програмування – це теоретична й практична діяльність по створенню програмного забезпечення. Теоретичні питання програмування стосуються розробки нових мов програмування, розробки способів доказів правильності програм, мінімізації складності та ін. Практика програмування займається розробкою трансляторів, зручного середовища програмування, конкретних баз даних та систем керування, проектування, навчання і т.д.

В наш час активно використовуються інтегровані середовища розробки, що включають в свій склад редактор для введення і редагування текстів програм, відладчики для пошуку і усунення помилок, транслятори з різних мов програмування, компонувальники для збірки програми з декількох модулів і інші службові модулі.

Мови високого рівня наближені до природних понять. Ці мови є машинно-незалежними. Для забезпечення якісного програмування існує технологія *структурного програмування*.

Згідно з маркетинговими дослідженнями, на початок 2018 року домінуючою настільною операційною системою є Microsoft Windows з часткою ринку близько 83,3%. Mac OS від Apple Inc. посідає друге місце (11,2%), а різновиди Linux перебувають на третій позиції (1,55%).

Microsoft позиціонує сьогодні свій пакет MS Office як комплексну платформу для створення бізнес-додатків, орієнтованих на широке коло завдань кінцевих користувачів. До складу сьогоднішніх версій MS Office входять і версії VBA, які застосовуються в багатьох офісних пакетах. Мова VBA, будучи уніфікованою мовою програмування для всіх додатків, спрощує створення програмних рішень.

Мова Visual Basic for Applications (Visual Basic для додатків, скорочено VBA) є основним інструментом офісного програмування, тобто програмування в пакетах MS Office, вона вбудована в лінійку продуктів Microsoft Office, а також в багато програмних пакетів, таких як AutoCAD, WordPerfect та ін.

Володіючи усіма рисами сучасних об'єктно-орієнтованих мов, VBA підтримує механізм візуального проектування форм, дозволяє включати в діалогові вікна і впроваджувати в документи елементи управління на базі ActiveX.

Мова С, створена Денисом Рітчі на початку 70-х років в Bell Laboratory американській корпорації AT&T, є одною з універсальних мов програмування і вважається мовою системного програмування, хоча вона, безумовно, зручна і при написанні прикладних програм.

Зручність мови С заснована на тому, що вона є одночасно і мовою високого рівня, яка має повний набір конструкцій структурного програмування, що підтримує модульність, блокову структуру програм, можливість роздільної компіляції модулів. У той же час мова С має набір низькорівневих засобів, що дозволяють мати зручний доступ до апаратних засобів комп'ютера, зокрема дістатися до кожного біта пам'яті.

Упродовж останніх років однією з найпоширеніших мов програмування, є мова С++, яка історично виникла як надбудова над мовою С. Мова С++ є мовою об'єктно-орієнтованого програмування, при створенні якої були використані поняття, які потім стали застосовуватися в мові С і увійшли в стандарт ANSI C (American National Standards Institute). Таким чином, мови С і С ++ надавали взаємний вплив один на одного.

Програмувати на С++ можна як для Windows, так і для Unix, причому для кожної з операційних систем існує значна кількість засобів розроблення: від компіляторів до потужних інтерактивних середовищ, таких як Borland C++ Builder, Microsoft Visual C++ , Visual Studio.NET.

Ми розглядаємо середовище Borland C ++ Builder як середовище, в якому розробка програм ведеться на основі об'єктно-орієнтованого програмування. Завдяки багатовіконному графічному інтерфейсу програми під час виконання можуть взаємодіяти з користувачем.

Навчальний посібник розкриває специфіку основних засобів використання мови VBA та можливостей потужної та гнучкої мови С++ у середовищі Builder.

1 ОСНОВИ VISUAL BASIC FOR APPLICATION

1.1 Основні властивості об'єктно-орієнтованого програмування

Об'єктно-орієнтоване програмування (ООП) – це модель програмування, яка базується на стверджені того, що програма – це сукупність об’єктів, які взаємодіють між собою; це методологія програмування, яка заснована на представленні програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію спадкування.

В даному визначенні основними є три частини: ООП використовує в якості базових елементів об'єкти; кожен об'єкт є екземпляром якогось певного класу; класи організовані ієрархічно. Програма буде об'єктно-орієнтованою тільки при дотриманні всіх трьох зазначених вимог. Кожен об’єкт в цій моделі є незалежним і здатним отримувати дані, обробляти їх та відправляти ці дані іншим об’єктам.

Основним поняттям ООП є об'єкт. Об'єкт можна визначити як певну сукупність даних (характеристик, атрибутів об'єкта) та методів роботи з ними. Для класифікації об'єктів у ООП використовують класи. Клас служить зразком для створення об'єкта, тобто об'єкт є нічим іншим, ніж копією класу.

Кожен об'єкт має процедури і функції (те що він уміє виконувати – завантажувати файл, відображати картинку і т.д.), які служать для роботи з даними об'єкта. Ці процедури і функції називаються методами.

Основу ООП складають такі основні концепції: інкапсуляція, успадкування, поліморфізм та модульність.

Інкапсуляція. Також відома як приховування даних. Зміст інкапсуляції полягає у приховуванні від зовнішнього користувача деталей реалізації об'єкта, замість цього надаючи інтерфейс взаємодії з ним.

Успадкування (спадкування). Це означає, що об'єкти (класи) можуть переймати деякі властивості у своїх прабатьків. Нюанси залежать від тієї мови, на якій пишеться програма, однак у будь-якому випадку це полягає у повторному використанні вже одного разу написаного коду. Підкласи успадковують атрибути та поведінку своїх батьківських класів, і можуть мати нові власні атрибути. Тобто утворюється ієрархія з класів, де від основного класу (так званого, предка) походять усі інші класи (рис. 1.1). Спадкування

являє собою процес, в ході якого один об'єкт набуває властивостей іншого об'єкта. Це дуже важливий процес, оскільки він забезпечує принцип ієрархічної класифікації.

Поліморфізм означає залежність поведінки від класу, якому ця поведінка притаманна, тобто, два або більше класів можуть реагувати по-різному на однакові повідомлення. Це спричинене зміною в одному з класів якогось методу (процедури, функції) шляхом запису іншого алгоритму. Як приклад, деяка комп'ютерна програма при натисканні клавіші Alt завершить роботу, інша ж програма після натискання кнопки Alt тільки відкриє меню даної програми.

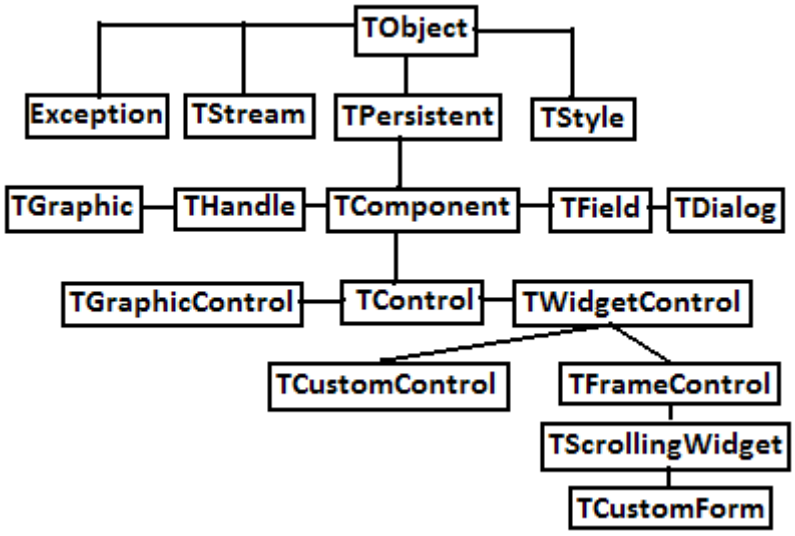


Рисунок 1.1 – Ієрархія класів у ООП

Слово «поліморфізм» можна перекласти як «багато форм». Тобто це можливість використання одного і того ж імені операції або методу до об'єктів різних класів, при цьому дії, що здійснюються з об'єктами, можуть істотно різнитися. Тому можна сказати, що в одного і того ж слова багато форм.

Модульність означає поділ програми на фрагменти, які компілюються окремо, але можуть встановлювати зв'язки з іншими модулями. У більшості мов, що підтримують принцип модульності як самостійну концепцію, інтерфейс модуля відділений від його реалізації.

У різних мовах програмування модульність підтримується по-різному. Наприклад, в C++ модулями є відокремлені компільовані файли. Для C/C++ традиційним є розташування інтерфейсної частини модулів в окремі файли з розширенням .h (так звані файли-заголовки). Реалізація, тобто текст модуля,

зберігається в файлах з розширенням .c. (в програмах на C++ часто використовуються розширення .cp і .cpp). Зв'язок між файлами оголошується директивою макропроцесора #include.

1.2 Проект VBA for Excel

Microsoft Visual Basic for Applications (VBA) – спеціальна надбудова, яка дозволяє вбудовувати спрощену версію мови Visual Basic в програми класу Microsoft Office. Мова VBA була розроблена компанією Microsoft, вона не є самостійною, а призначена для автоматизації процесів в пакеті MS Office. Однією з програм, в якій часто застосовується VBA є Microsoft Excel. Суть мови полягає в оперуванні об'єктами (що відносить його до об'єктно-орієнтованого програмування).

Об'єкти (Objects) VBA

Об'єкт – це елемент, структурна частка Excel, а саме: книга, лист, діапазон, комірка. Дані об'єкти мають ієрархію, тобто підкоряються один одному (рис.1.2).

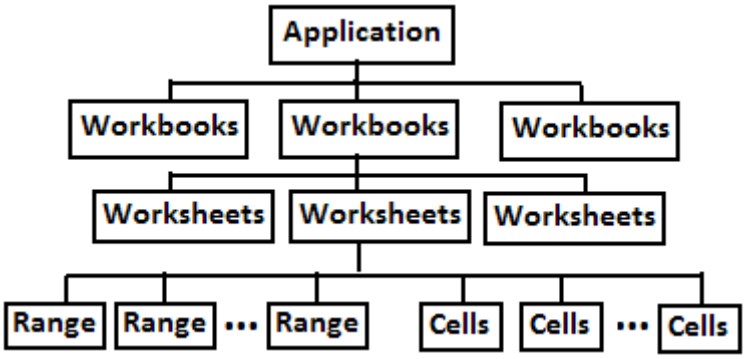


Рисунок 1.2 – Схематична структура ієрархії об'єктів Excel

Головний об'єкт – це Application, що відповідає самій програмі Excel. Далі слід Workbooks (Книга), Worksheets (Лист), Range (Діапазон), Cells (Комірка). Наприклад, щоб звернутися до діапазону комірок "A1:E10" на листі Excel, нам потрібно буде прописати наступний шлях з урахуванням ієрархії:

Application.Workbooks("Архив").Worksheets("Лист1").Range("A1:E10").

За замовчуванням проект VBA містить модуль Книги Excel (**ThisWorkbook**) і модулі трьох відкритих листів книги Excel. Форма **ThisWorkbook** являється головною формою Excel. Модуль ThisWorkbook є

головним модулем проекту. Даний модуль призначений для опису оточення проекту, а саме: оголошення додаткових бібліотек об'єктів (.olb), оголошення ActiveX управлінь (Active DLL) і динамічних бібліотек (DLL).

Кожна дочірня форма Excel (Лист) має власний модуль. Всі додаткові елементи управління, які установлені в листі Excel, належать даному листу. Програмний код обробки подій елементів управління реалізується в даному модулі.

Проект VBA може містити додаткові користувальницькі форми (UserForm). Додаткові форми в середовищі VBA основним чином застосовуються для створення власних діалогових вікон. Включати в проект додаткові модулі доцільно у випадках використання методів, які вироблені безпосередньо користувачем.

1.3 Інтерфейс Visual Basic for Applications

Запуск редактора VBA із середовища Excel можливий наступними способами:

- за допомогою гарячих клавіш Alt+F11;
- вкладка «Разработчик» → Елементи ActiveX → Редактор Visual Basic; для версії MS Office 2003: пункт меню Сервіс →Макрос → Редактор Visual Basic;
- панель інструментів Visual Basic → Редактор Visual Basic;
- панель інструментів Елементи управління → Початковий текст.

Повернення з редактора Visual Basic (VB) в робочу книгу Excel здійснюється за допомогою гарячих клавіш Alt+F6, або кнопкою на панелі інструментів Standard→View Microsoft Excel.

Головне вікно редактора VBA (рис.1.3) містить:

- системне меню (1);
- панель управління (2), що містить кнопки команд, які дублюють основні команди системного меню;
- вікно проекту – Project Explorer (3);
- вікно редактора властивостей об'єктів – Properties Window (4);
- вікно редактора коду – Code, в якому набирається програма (5);
- за потреби додається *форма* – UserForm (6).

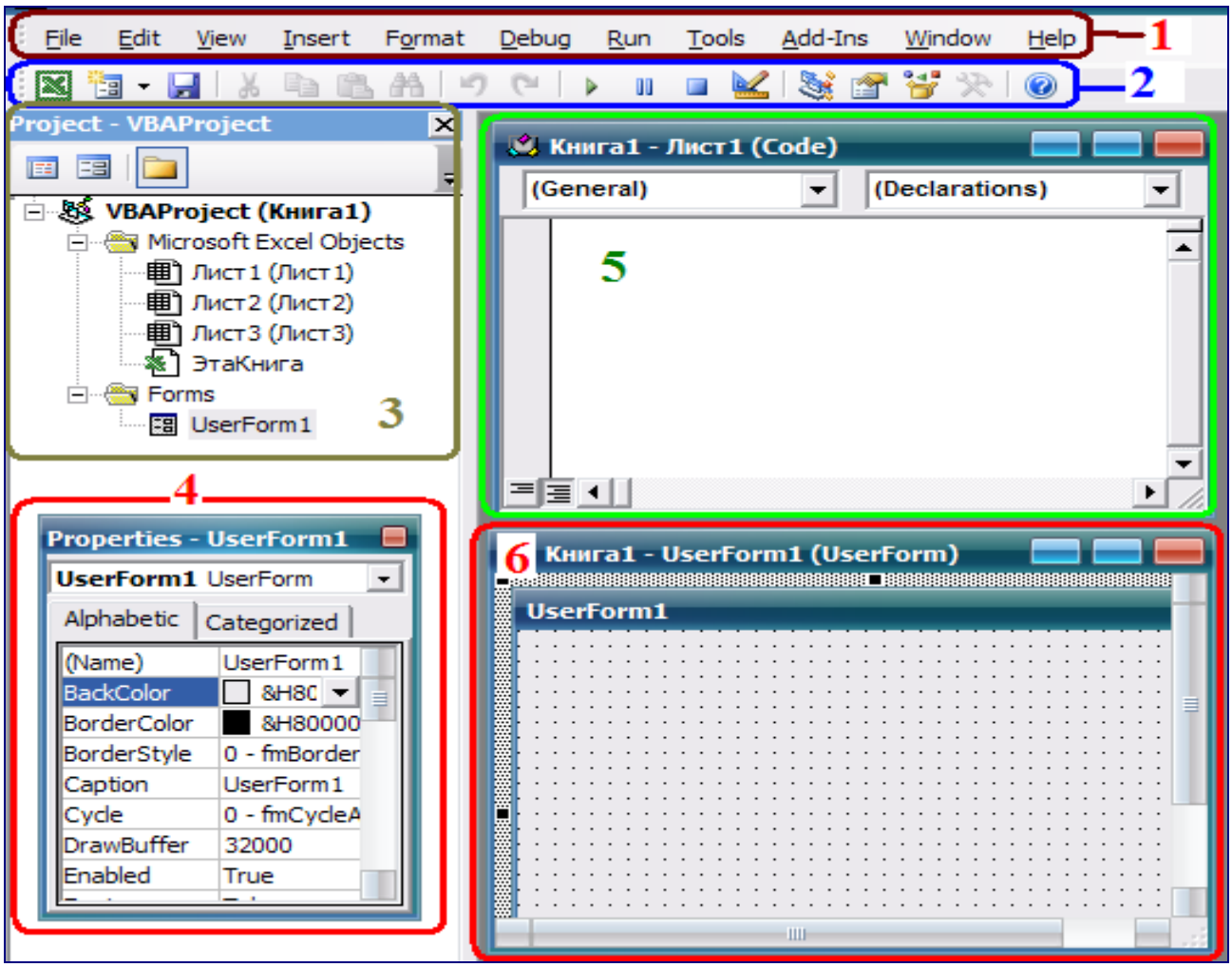


Рисунок 1.3 – Вікно середовища VBA

Доступ до вікна проекту і вікна редактора властивостей об'єктів здійснюється командами пункту меню *View → Project Explorer, View → Properties Window* відповідно.

Доступ до вікна редактора коду здійснюється командою пункту меню *View → Code,* за умови вибору модуля у вікні проекту, або подвійним клацанням миші над обраним об'єктом у вікні проекту.

Основними командами запуску та налагодження програмного коду є команди пункта меню *Run* (Запуск): *Run Sub* (Запуск програми), *Break* (Перервати), *Reset* (Скидання) і *Design Mode* (Конструктор). Дані команди дубльовані відповідними кнопками, розміщеними в панелі управління VBA. Команда *Конструктор* переводить VBA в режим дизайну.

Головний додаток, наприклад, Excel, спільно з VBA можуть знаходиться в двох станах, а саме в стані *виконання* (Run Time Mode) і в стані *конструктора* (Design Time Mode).

Стан **виконання** – це звичайний режим роботи Excel, при якому виконуються обчислення і інші активні дії з середовища Excel.

Стан **конструктора.** У цьому режимі розташовуються об'єкти, редагуються їх властивості, пишеться програмний код. Перехід в режим конструктора: панель інструментів *Елементи управління* → інструмент *Режим конструктора.*

1.4 Елементи управління

Елементи управління – це візуалізовані (видимі) об'єкти. Вони розташовані:

- у середовищі Excel на вкладці «Разработчик» → Елементи ActiveX;
- для версії MS Office 2003 на панелі інструментів Елементи управління;
- у середовищі VBA на панелі інструментів ToolBox.

Деякі з них:

- **Кнопка** (**CommandButton**) – для ініціалізації, закінчення або переривання яких-небудь дій;
- **Поле** (**TextBox**) – для відображення інформації, що вводиться користувачем, а також для відображення результатів обчислень;
- **Напис** (**Label**) – для відображення заголовків або короткого пояснювального тексту;
- **Рисунок** (**Image**) – призначений для виведення растрових зображень;
- **Полоса прокрутки (ScrollBar)** – дозволяє встановити числові значення, ґрунтуючись на положенні повзунка;
- **Рамка (Frame)** – декоративний елемент.

Елементи управління можуть бути розміщені:

- у середовищі Excel на робочому Листі (рис.1.4-а),
- у середовищі VBA – у вікні *форми* UserForm1 (рис.1.4-б).

Установка елемента управління у форму здійснюється при виборі відповідного об'єкта в палітрі й переміщенні його образа у форму. Для одержання доступу до властивостей елемента управління останній повинен бути обраний у формі. У вікні додатка Excel доступ до властивостей об'єкта здійснюється за допомогою кнопки «Властивості», яка розташована в панелі «Елементи управління».

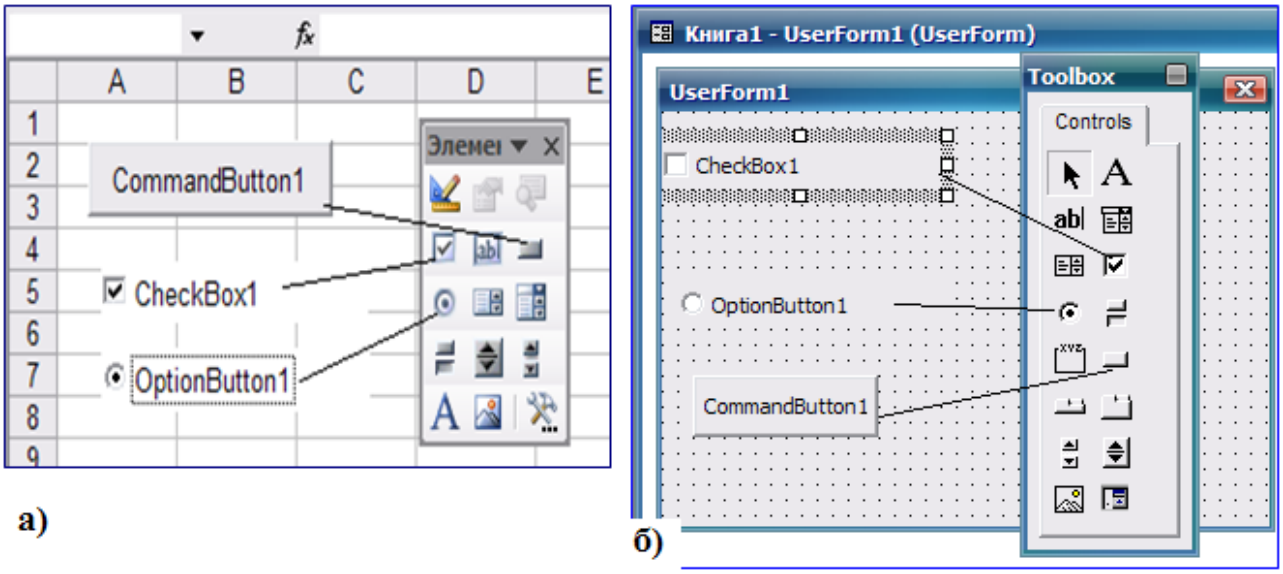


Рисунок 1.4 – Елементів управління: а) – на листі Excel, б) – на формі

Додавання *форми* в проект може бути здійснено наступним чином:

- перейти в редактор VB;
- пункт меню *Insert* → *UserForm*.

Основні властивості елементів управління

Основною властивістю будь-якого об'єкта є властивість *Name*. Властивість *Name* – це ідентифікатор об'єкта, тому не може існувати два і більше об'єктів з однаковим ім'ям. Властивість *Name* можна редагувати (змінювати) тільки в період оформлення програми. Як правило, об'єкту привласнюють унікальне ім'я, що відбиває фізичну сутність або призначення об'єкта.

Елементи управління – це видимі об'єкти, що мають такі основні властивості:

- Height – висота об'єкта;
- Width – ширина об'єкта;
- Left – розташування об'єкта ліворуч;
- Top – розташування праворуч;
- Caption – заголовок (напис) об’єкта;
- BackColor – колір фону об'єкта;
- BackStyle – зовнішній вигляд (стиль накреслення);
- ForeColor – колір напису (колір переднього плану).
- Picture – завантаження картинки;

За видимість об'єкта відповідає властивість *Visible*, а за доступ до управління – властивість *Enabled*.

Виклик вікна властивостей з середовища VBA здійснюється пунктом меню *View* → *Properties*.

Події й обробники подій елементів управління

Існує два види подій – це події, обумовлені діями користувача і події, що виникають в процесі виконання програми (програмно-керовані події). До подій користувача належать події миші і клавіатури. Всі інші події є програмно-керованими.

Операційна система Windows є системою, яка реагує на виникаючі події, тому, кожна подія має бути оброблена. З цією метою в VBA зумовлені спеціальні процедури обробки подій, заголовки яких називаються *обробниками подій*.

Найпростішим і найбільш часто використовуваним оброблювачем подій є процедура обробки події *клацання по об'єкту* – Click, наприклад:

```
Private Sub CommandButton1_Click()
 ' рядки програми, написані користувачем
End Sub
```

Як видно з конструкції процедури, вона не має ніяких параметрів, а просто передає управління програмному коду, написаному користувачем. Більш складні обробники подій передають в програмний код користувача один або кілька параметрів, які необхідні для реалізації управління. Наприклад:

```
Private Sub Image1_MouseMove (ByVal Button As Integer, ByVal Shift As
Integer, ByVal X As Single, ByVal Y As Single)
        'Рядки програми, написані користувачем
End Sub
```

Розглянемо призначення деяких існуючих оброблювачів подій користувальницької форми (UserForm).

UserForm_Activate – використовується для ініціалізації полів, загально-доступних змінних або установки значень властивостей об'єктів у момент, коли форма стає активною.

UserForm_Click – використовується для будь-яких дій при виконанні кліка мишею по формі.

UserForm_Deactivate – використовується для звільнення пам'яті й інших дій у період, коли форма стає неактивною.

UserForm_Initialize – використовується для ініціалізації полів, загальнодоступних змінних або установки значень властивостей об'єктів у момент, коли образ форми створюється в оперативній пам'яті.

Крім прихованих методів, що реалізують управління властивостями і подіями об'єктів, в VBA є доступні користувачеві методи елементів управління. Розглянемо деякі з них.

За допомогою методу *Show* користувач може вивести форму на екран:

UserForm.Show

За допомогою методу *Hide* користувач може приховати форму:

UserForm. Hide

За допомогою методу *PrintForm* користувач може вивести на друк образ форми з встановленими в ній елементами управління:

UserForm1.PrintForm

1.5 Доступ до комірок листа Excel. Форматування комірок

Введення даних в комірку Листа Excel здійснюється за допомогою властивостей об'єктів Cells (комірка) і Range (діапазон комірок).

Доступ до комірки за допомогою властивості Cells здійснюється через індекси R (Row – рядок) і C (Column – стовпець). Для того, щоб вибрати деяку комірку на листі Excel, досить у властивості Cells вказати її RC адресу і вибрати властивість Select.

Застосування властивостей Range і Cells в практичній роботі має ряд особливостей. Властивість Range доцільно застосовувати у випадках, коли наперед відомо (визначено), в яку комірку необхідно записати деякі дані або з якої комірки вибрати необхідні дані. Властивість Cells зручно використовувати для введення-виведення масивів даних або коли адреса комірки обчислюється.

Крім введення і виведення даних можна програмно оформити зовнішній вигляд комірки і діапазону комірок. Оформлення зовнішнього вигляду здійснюється за допомогою властивості *Interior* (інтер'єр) і вкладених в нього властивостей *Color* (колір), *ColorIndex* (індекс кольору), *Pattern* (візерунок) та інші. Ці параметри мають зумовлені значення.

Властивості *Color* можна привласнити значення кольору за допомогою символічного імені, або відповідного бінарного коду в шістнадцятковому представленні:

vbBlack – &H000000 – чорний; vbRed – &H0000FF – червоний;
vbGreen – &H00FF00 – зелений; vbYellow – &H00FFFF – жовтий;
vbBlue – &HFF0000 – синій; vbMagenta – &HFF00FF – рожевий;
vbCyan – &HFFFF00 – блакитний; vbWhite – &HFFFFFF – білий.

Наприклад, значення кольору задається безпосередньо і в шістнадцятирічному коді: Range("A1: A10"). Interior.Color = vbGreen
Range("B1:B10").Interior.Color = &HFF0000

У VBA клітина A2, як об'єкт, може бути оформлена двома варіантами: Range ("A2") або Cells(2,1), де 2 – номер рядка, 1 – номер стовпця.

Приклади.

```
Cells(2,1) = 14.6                'в клітину A2 заноситься число 14,6;
Range("A1") = Sin(0.5) 'в клітину A1 заноситься результат обчислення формули;
Range("A3") = Range("A1")*Cells(2,1)^2
```

Текст, що йде за символом ('), ігнорується компілятором і є коментарем.

Для оформлення зовнішнього вигляду клітин використовуються властивості **Interior** (інтер'єр), **Font** (шрифт) і вкладені в них властивості:

Color – колір; **Pattern** – візерунок;
Size – розмір; **Bold** – жирний;
Italic – курсив.

Приклади.

```
Range("A6") = "текст"                'текст, що вводиться, повинен бути у лапках
Range("A6").Interior.Color = vbGreen  'заливка діапазону зеленим кольором
Range("A6").Font.Color = vbBlue       'колір шрифту блакитний
Range("A6").Font.Bold=True            'шрифт жирний
```

2 ОСНОВНІ КОНСТРУКЦІЇ МОВИ VBA

2.1 Засоби мови VBA

Засобами мови VBA створюють проекти, які складаються з багатьох файлів, у середовищі певного офісного додатку. Автоматизація програмування

досягається завдяки тому, що є можливість використання готових об'єктів та їх налаштування під свої потреби.

У VBA користувач визначає імена *змінних, функцій, процедур, типів, констант та інших об'єктів.* У VBA існують такі обмеження на імена:

- довжина імені не повинна перевищувати 255 символів;
- в імені не може бути крапок, пробілів і символів: %, #, @, $, !, &;
- ім'я може містити будь-яку комбінацію букв, цифр, символів, які починаються з букви;
- не рекомендується вживати імена, які співпадають з ключовими словами VBA і іменами вбудованих функцій і процедур;
- імена повинні бути смисловими.

Змінні

Змінні використовуються для тимчасового зберігання даних в оперативній пам'яті, тобто вони ідентифікують області в пам'яті, де зберігається деяка інформація.

Типи даних

Типи даних відносяться до фундаментальних понять будь-якої мови програмування**.** Тип даних визначає множину допустимих значень, які може приймати вказана змінна. В Visual Basic for Application використовуються наступні типи даних:

Boolean логічні значення;

Integer цілий тип;

Single тип з плаваючою точкою звичайної точності;

Double тип з плаваючою точкою подвійної точності;

String тип рядок;

Variant тип, використовуваний за умовчанням, основний тип даних середовища VBA.

Тип даних Variant являється особливим. Змінні цього типу можуть містити будь-які дані, за винятком тих типів даних, котрі обумовлені користувачем. Змінна типу Variant може також містити спеціальні значення **Empty** (порожньо), **Error** (помилка), **Nothing** (нічого) і **Null** (не дійсний).

Користувач має можливість використати тип Variant замість будь-якого типу даних, щоб забезпечити більшу гнучкість при обробці даних.

Кожна змінна має бути оголошена до її застосування. Це здійснюється за допомогою оператора оголошення змінних **Dim**.

Приклад. Оголошення простих змінних.

```
Dim a, c              'за умовчанням тип змінних Variant
a = 100.5: c = 0      'декілька операторів в одному рядку відділяються знаком ":".
```

Локальна змінна може бути доступна тільки в тій процедурі, в якій вона оголошена. *Глобальна* змінна доступна на рівні модуля.

Змінна типу *Variant* може бути переоб'явлена в одновимірний масив, використовуючи функцію **Array**. Наприклад:

```
Dim a
a = Array(10, 20, 30)          'нумерація елементів йде з нуля
Range ("A1")=a(0)              'в клітині A1 відобразиться 10
Range ("A2")=a(2)              'в клітині  A2 відобразиться 30
```

Масиви

Масив – це набір елементів однакового типу, які мають загальне ім'я. А звернення до цих елементів відбувається по їх індексу. По розміру масиви ділять на два види:

- статичні – кількість елементів масиву оголошується на етапі розробки і не змінюється в процесі виконання програми;
- динамічні – число елементів і розмірність змінюється в процесі роботи програми.

Статичні масиви оголошуються із зазначенням верхньої і нижньої межі:

```
Dim myArray(0 to 49) as Variant
```

Оголосити масив також можна, вказавши тільки кількість елементів:

```
Dim myArray(50) as Variant
```

Для оголошення багатовимірних масивів використовується запис наступного вигляду:

```
Dim Matrix2D(0 to 9, 0 to 9) as Variant
Dim Matrix3D(10, 10, 10) as Variant                    'тривимірний масив'
```

Приклад 1. Оголошення та ініціалізація одновимірного масиву з трьох елементів:

```
Dim A(2)                    'верхнє значення індексу дорівнює 2
A(0)= 3.6: A(1)= 4.82
A(2)= – 12.7
```

Приклад 2. Оголошення та ініціалізація двовимірного масиву:

```
Dim B(1,1)
B(0,0)=2:   B(0,1)=4
B(1,0)=1:   B(1,1)=6
```

Приклад 3. Одномірний масив цілих чисел з 12 елементів.

```
Dim A(12) As Integer
```

За замовчуванням перший елемент масиву – A(0), а останній – A(11). В цьому випадку говорять, що 0 – базовий індекс. Його можна змінити, написавши інструкцію Option Base 1. Тоді першим елементом масиву буде A(1).

```
Dim B(3, 3) As Single      'двовимірний масив (матриця) дійсних чисел.
```

Динамічні масиви. В цьому випадку при об’явленні не треба вказувати розмірність, наприклад:

```
Dim R() As Single
```

Потім в програмі необхідно обчислити необхідний розмір масиву в деякій змінній, і змінити розмір динамічного масиву за допомогою інструкції ReDim.

Приклад 1. Встановлення меж динамічного масиву від 1 до 10.

```
ReDim R(1 to 10)
```

Приклад 2. Використання інструкції ReDim.

```
Dim B( ) As String
N = 2                       ‘обчислюється розмір масиву
M = 2
ReDim B(N, M)               ‘зміна розміру масиву
N = 3
M = 3
ReDim(N, M)
```

Операції VBA. В програмах на VBA можна використовувати стандартний набір операцій над даними. Існує три типи операцій:

- математичні (виконуються над числами і їх результатом є числа);

- відношення (виконуються не лише над числами і їх результатом є логічні значення, наприклад x < y);
- логічні (застосовуються для логічних виразів і їх результатом є логічні вирази, наприклад Not x And y).

Математичні операції:

+, -, *, /, \ (цілочисельне ділення),

Mod (залишок від ділення);

^ (піднесення до степені).

Операції відношення: <, >, <=, >=, <> (не рівно), =.

Логічні операції: виконуються з даними логічного типу (boolean).

Результат обчислення логічної операції, в залежності від значень операндів X і Y, наведені у таблиці істинності:

x	**y**	**AND**	**OR**	**NOT x**
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

1 – істина; 0 – неправда **And (**логічне множення**);**

Or (логічне складання**);**

Not (логічне заперечення**).**

Пріоритет операцій: Not, And, Or.

Приклад. Нехай x(i) – елемент масиву, тоді:

	запис у математиці:	*запис на мові VBA:*
1)	$x_i \in [-1, 1]$	x(i)>= - 1 And x(i)< =1
2)	$x_i \in (-\infty;-1] \cup [1;\infty)$	x(i)<=-1 OR x(i)>=1

Інші операції. [Рядок1] & [Рядок2] – зчеплення рядків.

Вбудовані функції. В VBA є великий набір вбудованих функцій (табл.2.1) і процедур, використання яких суттєво спрощує програмування. Ці функції можна розділити на наступні основні категорії: математичні функції, функції перевірки типів, функції перетворення форматів, функції обробки рядків, функції дати і часу.

Таблиця 2.1 – Математичні функції VBA

Функція	Опис
Abs	модуль (абсолютна величина) указаного числа
Atn	арктангенс числа
Cos	повертає значення косинусу кута
Exp	експонента, тобто піднесення числа **e** до вказаного степеня
Log	натуральний логарифм
Int	повертає значення аргументу, що утримує цілу частину числа, котре менше, або дорівнює указаному
Rnd	повертає випадкове число з інтервалу [0;1)
Sin	повертає значення синусу кута
Sgn	знак числа
Sqr	квадратний корінь указаного числа
Tan	тангенс кута
Mod	залишок від ділення

2.2 Обчислювальні процеси у VBA

Розрізняють три базові алгоритмічні структури:

- лінійна структура (дії виконуються послідовна, одна за одною);
- розгалужена структура (результат обчислення залежить від певної умови);
- циклічна структура (в обчислювальному процесі присутній повтор однотипних операцій).

Обчислювальний процес, що розгалужується

Даний процес потребує наявності операторів умовного переходу IF, Then, Else. Існує декілька варіантів використання цих операторів.

1. **If умова Then оператори**

Якщо умова набуває значення *істина*, то виконуються оператори, що йдуть за *Then,* інакше – оператор, що йде за *IF*.

2. **If умова Then оператори1 Else оператори2**

Якщо умова набуває значення *істина*, то виконуються *оператори1,* що йдуть за *Then***,** інакше **–** *оператори2,* що йдуть за ***Else*.**

3. Блок **If:**

If умова Then

операторі1

[**Else**

операторі2]

End If

Група **[Else** операторі2**]** може бути відсутня**.** Блок **If** виконується аналогічно випадкам а) і б), використовується для зручності запису.

Приклад. Обчислити значення функції

$$z=\begin{cases} a+b, & при \quad a>b \\ a-b, & при \quad a<b \\ a*b, & при \quad a=b \end{cases}.$$

Нехай значення *a i b* знаходяться в клітинах A2 і B2 відповідно.

Процедура на мові VBA має вигляд:

```
Sub f1()
  Dim a, b, z                          'оголошення змінних
  a = Range("A2") : b = Range("B2")
  If a > b Then z = a + b
  If a < b Then z = a - b
  If a = b Then z = a * b
  Range("C1") = z                      'результат поміщається в клітину C2
End Sub
```

Приклад. Знайти дійсні корені квадратного рівняння $ax^2+bx+c=0$. Значення a, b, c вводяться в VBA з клавіатури, результат виводиться в комірки Листа.

В даному прикладі використовується функція *InputBox*, яка виводить на екран вікно, що містить повідомлення і поле введення. Вона повертає значення типу *String*. Для перетворення текстового типу в числовій використовується функція *Val*, наприклад:

```
a = val(InputBox("a"))
```

Порядок дій:

1. У режимі *Конструктор* розмістити два об'єкти *Кнопка* для запуску проекту і для очищення діапазону (рис.2.1), дати їм заголовки *Розв'язок рівняння* і *ОЧИСТИТИ* відповідно.

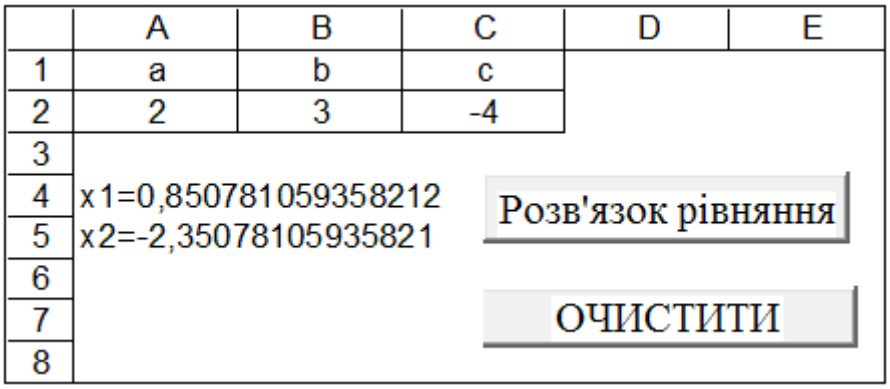


Рисунок 2.1 – Оформлення завдання

2. Обробник події *Click* кнопки *Розв'язок рівняння* має вигляд:

```
Private Sub CommandButton1_Click()
Dim a, b, c, x1, x2, d
a = Val(InputBox("a"))          'значення змінної a, що вводиться з клавіатури
b = Val(InputBox("b"))          'значення змінної b, що вводиться з клавіатури
c = Val(InputBox("c"))          'значення змінної c, що вводиться з клавіатури
Range("A2") = a                 'запис значення змінної a в комірку Листа
Range("B2") = b
Range("C2") = c
d = b ^ 2 - 4 * a * c                  ' обчислення дискримінанту
If d >= 0 Then                         ' перевірка умови
  x1=(-b+sqrt(d)/(2 * a)               ' знаходження коренів
  x2=(-b-sqrt(d)/ (2 * a)
  Range("A4") = "x 1=" & x1            ' виведення рішення  в комірку
  Range("A5") = "x 2=" & x2
  Else
  Range("A4") = "дійсних коренів немає"
End If
End Sub
```

3. Обробник події *Click* кнопки *ОЧИСТИТИ* має вигляд:

```
Private Sub CommandButton2_Click()
   Range("A4: A5").ClearContents
End Sub
```

4. Повернутися в Excel, вийти з режиму *конструктор*, запустити проект на виконання (RUN або F5). Перевірити роботу проекту кліком миші по кнопках.

Циклічний обчислювальний процес

Під час роботи з масивами їх обробку значно спрощує використання циклів, для перебору індексів масиву. Виконання циклічного обчислювального процесу забезпечується багатьма операторами. Розглянемо роботу операторів циклу FOR … NEXT.

Загальний вигляд оператора циклу:

FOR *змінна* **= A1 TO A2 [step A3]**
операторі
NEXT [змінна]

де *змінна* – параметр циклу;

A1 – початкове значення параметра циклу;

A2 – кінцеве значення параметра циклу;

A3 – крок зміни параметра циклу (якщо крок дорівнює 1, то може бути відсутнім).

Оператор FOR ... NEXT повторює виконання групи операторів, поки параметр циклу змінюється від початкового значення до кінцевого значення з зазначеним кроком. У наступному прикладі за допомогою оператора циклу знаходиться сума елементів масиву.

Приклад. Оператор циклу з одиничним кроком зміни параметра циклу.

```
Private Sub CommandButton1_Click()
Dim A
A=Array(1, 4, 12, 23, 34, 3, 23)        'змінна А переоб'являється в масив
S=0
For i=Lbound(A) To Ubound(A)
  s=s+A(i)                              'накопичення суми
```

```
Next
Range("A2")=s
End Sub
```

Функція Lbound() винаходить межу масиву, Lbound(A) – нижня межа масиву А; Ubound(A) – верхня межа масиву А.

Обробка масивів

Одновимірна масив – це найпростіший варіант масиву, який використовує звичайний список даних. Наприклад: коло, еліпс, трикутник, квадрат, прямокутник, ромб. Це строковий масив, що складається з 6 елементів. Привласнемо його змінній *myArray* за допомогою функції *Array*:

```
Dim myArray As Variant
myArray = Array("коло", "еліпс", "трикутник", "квадрат", "прямокутник",
"ромб")
```

myArray(3) – цьому елементу нашого масиву відповідає "квадрат" (тому, що за замовчуванням нумерація елементів масиву починається з 0).

Приклад обробки одновимірного масиву.

Заданий масив чисел X=(-17; 8; -0,3; 6; -9; 55; 0; 78; -9: -45). Знайти суму від'ємних елементів (нехай такі елементи існують) та кількість інших елементів масиву.

Розглянемо випадок, коли масив вводиться у програмі і виводиться в комірки Листа Excel.

Порядок дій:

1. У режимі *Конструктор* встановити об'єкт CommandButton1, дати йому заголовок *ОБЧИСЛИТИ* (рис.2.2).

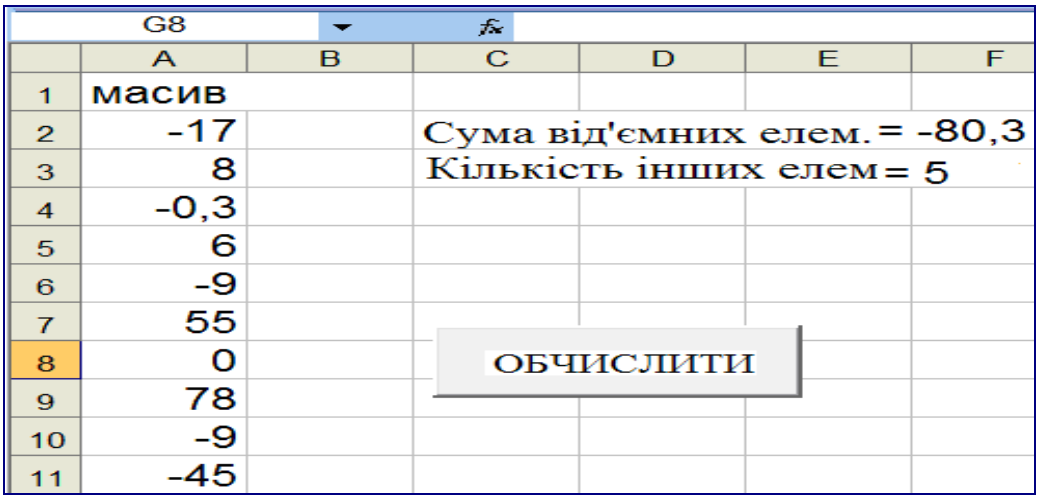


Рисунок 2.2 – Оформлення завдання

2. Обробник подій *Click* кнопки *ОБЧИСЛИТИ* буде мати вигляд:

```
Private Sub CommandButton1_Click()
   Dim x, k, s, i
   x = Array(-17, 8, -0.3, 6, -9, 55, 0,78, -9, -45)  'змінна X стає масивом
   s = 0: k = 0                          'сума і кількість елементів
   For i = 0 To 9                        'нумерація елементів масиву з 0
   Cells(i + 2, 1) = x(i)                'виведення елементів масиву в 1-й стовпець
   If x(i)< 0 Then
   s = s + x(i)                          'накопичення суми
   Else
   k = k + 1
   End If
   Next
   Range("c2") = "Сума від'ємних елементів = " & s
   Range("c3") = "кількість інших елементів = " & k
End Sub
```

Генерація масиву випадкових чисел

Функція **Rnd()** повертає випадкове число з діапазону [0;1). Можливі різні способи використання цієї функції, наприклад:

100***Rnd()**	повертає випадкове число з діапазону [0;100);
a+(в-a)***Rnd()**	повертає випадкове число з діапазону [a;в);
Int(100***Rnd())**	повертає ціле випадкове число з діапазону [0;99];
Randomize	ініціалізує генератор випадкових чисел.

Приклад коду:

```
Sub Test ()
   Dim MyValue
   Randomize
   MyValue = Int (6 * Rnd()) + 1)          'повертає випадкове число від 1 до 6
End Sub
```

В даному прикладі перед викликом функції *Rnd* використовується оператор *Randomize* без аргументу для ініціалізації генератора випадкових чисел значенням, що повертається системним таймером.

Приклад. Згенерувати масив випадкових чисел з 10 елементів.

For 0 = 1 To 9

```
  x(i) = 100*Rnd() – 50          'випадкові числа з діапазону [-50;50)
Next
```

Динамічні масиви

Якщо розмірність масиву в програмі не постійна, то такий масив називається *динамічним*. Динамічні масиви оголошуються за допомогою оператора **ReDim.** Програма буде універсальною, якщо працюватиме для масиву будь-якої розмірності.

Приклад. Створити процедуру для генерування одновимірного масиву цілих випадкових чисел будь-якої розмірності. Знайти середнє геометричне позитивних елементів цього масиву.

Порядок дій:

У режимі *Конструктор* встановити елемент управління *CommandButton1*, дати йому заголовок О*БЧИСЛИТИ* (рис.2.3)

Нехай розмірність масиву знаходиться в комірці F1.

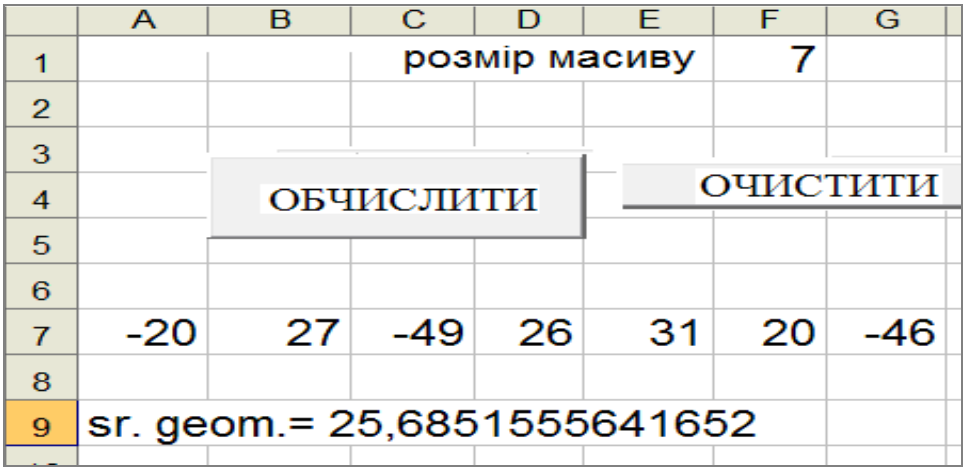


Рисунок 2.3 – Оформлення завдання

Обробник події *Click* кнопки *ОБЧИСЛИТИ* має вигляд:

```
Private Sub CommandButton1_Click()
  Dim p, k, n, i, sg
  n = Range("F7")                 ‘розмірність масиву
  ReDim x(n – 1)                  ‘оголошення масиву, нумерація елементів з 0
  Randomize
  For i = 0 To n - 1
  x(i) = int(100 * Rnd() – 50)                'діапазон [-50;50]
  Cells(7, i + 1) = x(i)                      ‘вивід елементів масиву в 7 рядок
  Next
```

```
  p = 1: k = 0                          ‘сума і кількість елементів масиву
  For i = 0 To n - 1
    If x(i)> 0 Then
    p = p * x(i) : k = k + 1
    End If
  Next
  Sg=p^(1/k)                            ‘середнє геометричне елементів масиву
  Range("a9") = "sr. geom.= " & sg
End Sub
```

Обробник події *Click* кнопки *ОЧИСТИТИ* має програмний код:

```
Private Sub CommandButton2_Click()
  Range("A7: G9").ClearContents
End Sub
```

Двовимірні масиви

Приклад 1. Заповнити масив X(3,4) цілими випадковими числами, вивести його в комірки Листа Excel.

```
For i = 0 To 2
For j = 0 To 3
x(i, j) = int(100 * Rnd() – 50)   'цілі випадкові числа з діапазону [- 50;50)
Cells(i + 1, j + 1) = x(i, j)
Next
Next
```

Приклад 2. Згенерувати масив цілих випадкових чисел розмірністю 3*3. Знайти суму елементів в кожному рядку масиву.

Порядок дій:

1. Перейти в режим *Конструктора,* встановити елементи управління *CommandButton1* і *CommandButton2,* дати їм заголовки (властивість ***Caption***) *Згенерувати масив та обчислити* та *Очистити* відповідно.
2. Обробник події *Click* кнопці *Згенерувати масив та обчислити* має вигляд*:*

```
Private Sub CommandButton1_Click()
  Dim i, j, x(2, 2)                 'оголошення змінних і масиву розмірністю 3*3
  Randomize
  For i = 0 To 2
```

```
  For j = 0 To 2
    x(i, j) = Int(100 * Rnd() – 50)       'цілі випадкові числа з діапазону [- 50;49]
    Cells(i+5, j+1) = x(i, j)
  Next j
  Next i
  For i = 0 To 2
      s = 0
      For j = 0 To 2
        s = s + x(i, j)                          'накопичення суми
      Next j
      Cells(i+5, 4) = s                          'вивод суми
  Next i
End Sub
```

3. Обробник події *Click* кнопки *ОЧИСТИТИ* має вигляд:

```
Private Sub CommandButton2_Click()
  Range("a5: d7").ClearContents
End Sub
```

Результат роботи проекту надано на рисунку 2.4.

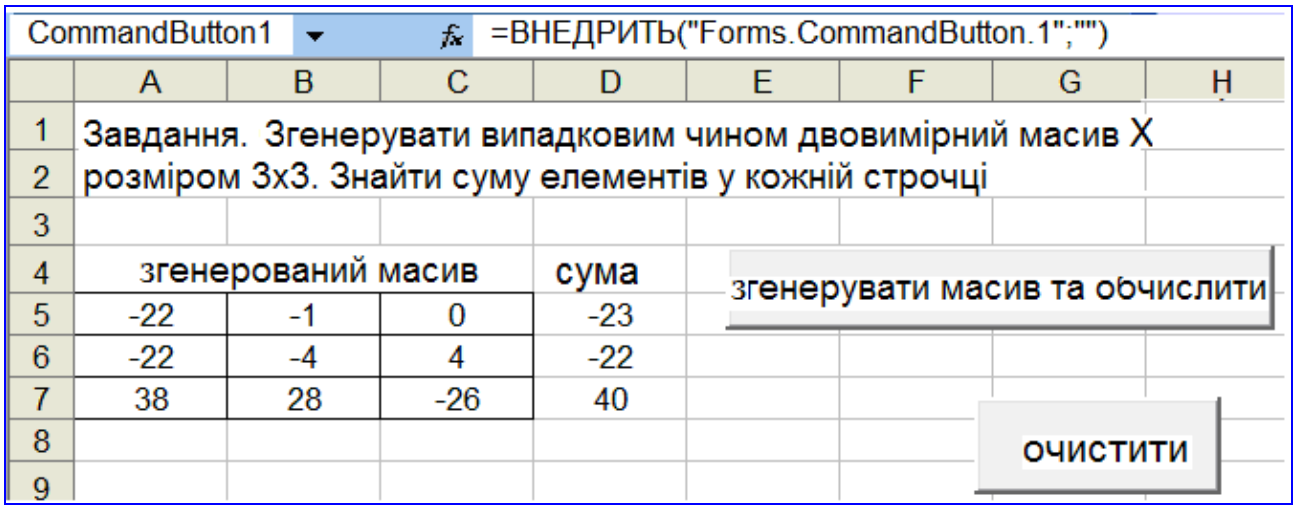


Рисунок 2.4 – Результат роботи проекту

2.3 Індивідуальні завдання та приклад їх виконання

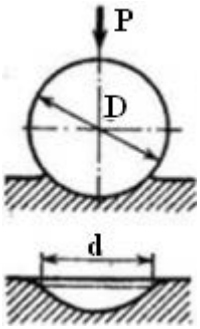
Приклад виконання. При визначенні твердості (НВ) металу за методом Брінелля застосовують формулу:

$$HB = \frac{P}{F},$$

де НВ – твердість металу;

P – навантаження на кульку, дин. P=3000 дин;

F – поверхня відбитка, мм^2;



$$F = \frac{\pi D}{2}(D - \sqrt{D^2 - d^2}),$$

де D – діаметр кульки, яка вдавлюється, мм. D=15 мм;

d – діаметр відбитку, мм.

У середовищі VBA створити програму, яка обчислює значення твердості металу для різних показників діаметра відбитку d. Діаметр відбитку d може змінюватися від $d_п$=0,75 мм до $d_К$ = 2,5 мм з кроком Δd = 0,25 мм. Рішення задачі наглядно оформити на листі Excel.

Покрокове виконання завдання.

Заповнити комірки на листі Excel згідно з рисунком 2.5. Кнопки *Обчислення* і *Оновити* – це елементи ActivX CommandButton1 та CommandButton2 відповідно.

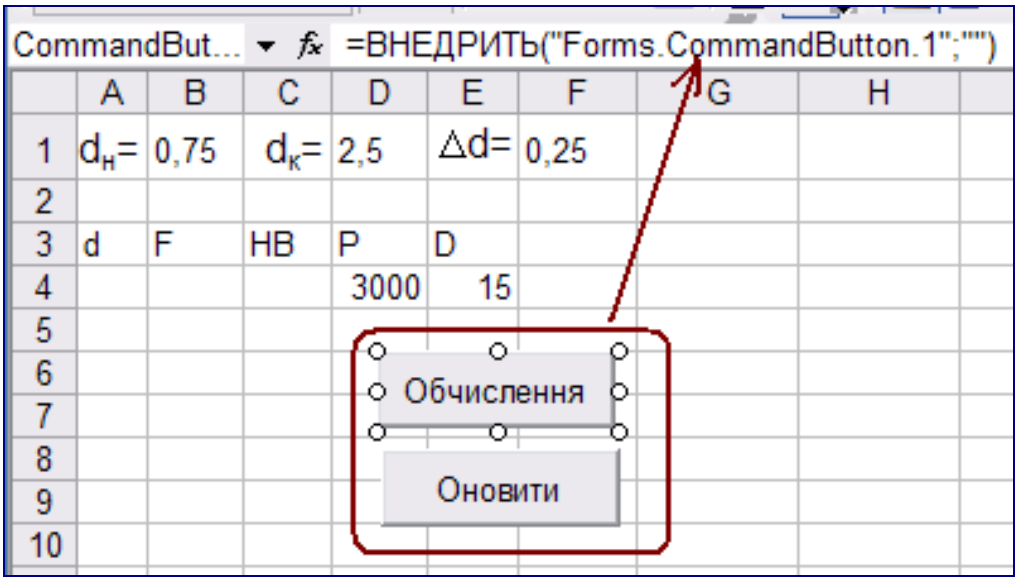


Рисунок 2.5 – Оформлення завдання

В режимі *Конструктор* виконати мишкою подвійний клік (від англ. Click – натискати) по кнопці CommandButton1 (кнопка *Обчислення*) і у обробнику подій, що відкрився, написати такий програмний код:

Private Sub CommandButton1_Click()

Dim i, j, d, diam(8), F(8), HB(8)

```
dn = Range("B1")
dk = Range("D1")
del_d = Range("F1")
P = Range("D4")
DD = Range("E4")
i = 1
For d = dn To dk Step del_d
diam(i) = d
F(i) = 3.14 * DD / 2 * (DD - Sqr(DD ^ 2 - d ^ 2))
HB(i) = P / F(i)
i = i + 1
Next
For i = 1 To 8
Cells(i + 3, 1) = diam(i)
Cells(i + 3, 2) = F(i)
Cells(i + 3, 3) = HB(i)
Next
```

End Sub

Розробити таблицю відповідностей:

Змінна	Позначення
diam	діаметр відбитку d
dn	початкове значення діаметра відбитку d_H
dk	кінцеве значення діаметра відбитку d_K
del_d	крок зміни значення діаметра відбитку Δd
P	навантаження на кульку
DD	діаметр кульки
HB	твердість металу
F	поверхня відбитка

В режимі *Конструктор* виконати мишкою подвійний клік по кнопці CommandButton2 (кнопка *Оновити*) і у обробнику подій, що відкрився написати такий програмний код:

```
Private Sub CommandButton2_Click()
Range("A4:C11").Select
```

```
Selection.Clear
Range("A1").Select
End Sub
```

Вимкнути режим *Конструктор* та натиснувши на кнопку *Обчислення*, отримати результат згідно з рисунком 2.6.

	A	B	C	D	E	F
1	$d_н$=	0,75	$d_к$=	2,5	Δd=	0,25
2						
3	d	F	HB	P	D	
4	0,75	0,441839	6789,806	3000	15	
5	1	0,785874	3817,405			
6	1,25	1,228699	2441,606			
7	1,5	1,770688	1694,257		Обчислення	
8	1,75	2,412299	1243,627			
9	2	3,154081	951,1487			
10	2,25	3,996672	750,6246		Оновити	
11	2,5	4,940803	607,1888			

Рисунок 2.6 – Результат виконання проекту

Варіанти індивідуальних завдань

1. Зміна напруження у виробі на стадії витримки після зварки підкоряється такому закону: $\sigma_t = \dfrac{\sigma_0}{\left(1+\dfrac{t}{t_0}\right)^{\beta}}$,

де σ_0 – початкове напруження до стадії витримки, $\frac{\hat{e}\tilde{a}-\tilde{n}}{\grave{i}\grave{i}}$;

t – час витримки, хв.; t_0 = 10 хв.;

β – час для різних сталей і температур, знаходиться у межах 0,88 ÷ 0,18.

У середовищі VBA створити програму, яка дозволить розрахувати зміну напруження у виробі в період часу від t_H до t_k з кроком Δt=1 хв., якщо β=0,1 та σ_0 приймає три значення 41,7; 30,6; 17,2.

Рішення задачі наглядно оформити на листі Excel.

2. Залежність кутової швидкості обертання ω маховика, лінійної швидкості V руху точок на ободі колеса та кінетичної енергії W_k маховика від зміни його кількості обертів n обчислюється за такими виразами:

$$\omega = \frac{\pi \cdot n}{30}; \qquad v=\omega \cdot R; \qquad W_k = \frac{mv^2}{2}.$$

У середовищі VBA створити програму, яка дозволить розрахувати та створити таблицю значень n, ω, v та W_k для маховика діаметром D=1,5 м та масою m=0,5 т, якщо кількість його обертів має такі значення: n= 50, 150, 200, 350, 550 та 900 об./хв.

Рішення задачі наглядно оформити на листі Excel.

3. Під час розрахунку теплового балансу доменних печей тепло, яке забирає вода охолодження, може бути обчислене як:

$$Q = K_1 \cdot (0.2n + 0.75 \cdot \sqrt[3]{V}),$$

де V – об'єм, $м^3$;

n – кількість повітряних фурм;

K_1 – коефіцієнт, який враховує сорт чавуну (переробний – 1; ливарний – 1,1; спеціальний – 1,2).

У середовищі VBA створити програму, яка дозволить розрахувати теплові втрати (Q), під час охолодження водою для переробного, ливарного та спеціального чавунів для печей різних об'ємів та відповідної кількості фурм:

V, $м^3$	1033	1300	1513	2000	2700	3200	5000
n	14	16	18	20	20	28	36

Рішення задачі наглядно оформити на листі Excel.

4. Секундний об'єм металу у кожній кліті стану безперервної прокатки розраховується як:

$$W=F_i \cdot V_i,$$

де F_i – площа поперечного перерізу труби під час виходу з **i**-тої кліті:

V_i – швидкість виходу труби з **i**-тої кліті.

У середовищі VBA створити програму, яка дозволить розрахувати секундний об'єм металу у кожній кліті для різних площ поперечного перерізу при різних швидкостях виходу труби з кліті. Значення F_i та V_i для кожної кліті наведено нижче:

№ кліті	1	2	3	4	5	6	7	8
Площа поперечного перерізу труби F_i, мм	2446	1642	1852	1022	846	758	716	712
Швидкість виходу труби із кліті V_i, м/с	0,76	1,1	1,39	1,86	2,24	2,52	2,64	2,83

Рішення задачі наглядно оформити на листі Excel.

5. Хід окислення металу виливки залежить від співвідношення об'ємів окалини V_{OK} і металу V_{ME}, визначається таким чином:

$$g = \begin{cases} k \cdot P \cdot \tau, & \text{êîëè } \dfrac{V_{ÎÊ}}{V_{ÌÅ}} \le 1 \\ \sqrt{k \cdot P \cdot \tau}, & \text{êîëè } \dfrac{V_{ÎÊ}}{V_{ÌÅ}} > 1 \end{cases},$$

де k – константа швидкості процесу, k=1,2;

P – тиск (концентрація кисню), P=28;

τ – час окислення, τ=2,5 хвилини.

У середовищі VBA створити програму, яка обчислює кількість продифундированої речовини **g** при наступній зміні співвідношення об'ємів:

V_{OK1}=10; V_{OK2}=20; V_{OK3}=30;

V_{ME1}=40; V_{ME2}=30; V_{ME3}=20.

Рішення задачі наглядно оформити на листі Excel.

6. Під час розрахунку значення твердості металу застосовують вираз:

$$F = \pi \cdot R \cdot (2R - \sqrt{4 \cdot R^2 - d^2}),$$

де HB – твердість металу;

P – навантаження на кульку, дин.;

F – поверхня відбитка, мм^2;

R – радіус кульки, яка вдавлюється, мм;

d – діаметр відбитку, мм.

У середовищі VBA створити програму, яка обчислює значення твердості металу для різних показників діаметра відбитку d. Діаметр відбитку d може змінюватися від $d_п$=0.80 мм до d_K = 3.5 мм з кроком Δd = 0.25 мм. P=2800 дин.; R=32 мм. Рішення задачі наглядно оформити на листі Excel.

7. Рівняння температурної залежності питомого електроопіру рідкого заліза має вигляд:

$$\rho = K_1 \cdot (1 + K_2 \cdot (t - K_3)),$$

де $К_1$, $К_2$, $К_3$ – коефіцієнти, значення яких змінюються в залежності від вмісту домішок та добавок в металі. Показники коефіцієнтів при різних концентраціях вуглецю в залізі:

0,05% $К_1$= 1351; $К_2$= $2{,}88\cdot10^{-4}$; $К_3$= 1535.

0,38% $К_1$= 1548; $К_2$= $3{,}30\cdot10^{-4}$; $К_3$= 1513.

1,26% $К_1$= 1464; $К_2$= $4{,}11\cdot10^{-4}$; $К_3$= 1443.

У середовищі VBA створити програму, яка обчислює питомий електроопір рідкого заліза при зміні температури t металу в межах від $t_{поч}$=1400 ^{0}C до $t_{кін}$=1600 ^{0}C з кроком Δt=50 ^{0}C для наведених значень коефіцієнтів $К_1$, $К_2$, $К_3$. Рішення задачі наглядно оформити на листі Excel.

8. Залежність ходу сильфону від прикладеної осевої сили розраховується як:

$$X = 0.32\cdot10^{-9}\cdot\frac{N_C\cdot D_C^2\cdot n\cdot\left(1-D_0/D_C\right)^3}{h_C^{2.5}\cdot\left(1+0.013\cdot l_{гоф}^2\right)},$$

де N_C – навантаження на сильфон, Н;

D_C та D_o – відповідно зовнішній та внутрішній діаметри сильфону, мм;

$l_{гоф}$ – крок гофри, мм.

У середовищі VBA створити програму, яка обчислює хід сильфону при зміні числа гофрів n від $n_{поч}$=5 до $n_{кін}$=50 з кроком Δn=5, коли N_C=300 Н, D_C=150 мм, h_C=0,3 мм, $l_{гоф}$=5 мм. В окремий стовпець вивести значення X та n, при яких хід сильфону X більше 5 мм. Рішення задачі наглядно оформити на листі Excel.

9. Необхідне зусилля затиску оправки в конічному гнізді шпинделя знаходять за виразом:

$$Q=1.75\cdot P\cdot(tg\varphi + tg\rho)\cdot(D+d)\cdot L+S\ ,$$

де P – питомий тиск на поверхні контакту, що дорівнює при чистій обробці 200 Н/см^2, при чорновій – 400 Н/см^2;

φ – кут між віссю конуса та його образуючою;

ρ – кут тертя у з'єднанні під час затиску оправки;

D та d – відповідно самий великий та самий маленький діаметри з'єднання;

L – довжина конічного з'єднання;

S – осьове зусилля різання, яке виштовхує оправку.

У середовищі VBA створити програму, яка обчислює залежність Q від зміни φ і ρ в градусах для чистової та чорнової обробки, якщо $10 \le \varphi \le 20$ з кроком 1, $0.5 \le \rho \le 0.75$ з кроком 0.1, коли D=90 мм, L= 200 мм і S=8 кг. Рішення задачі наглядно оформити на листі Excel.

10. В металургійних формах товщина шару затверділого металу змінюється в часі за лінійним законом:

$$\zeta = M \cdot t ,$$

де M,t – відповідно коефіцієнт та час затвердіння металу.

В цьому випадку коефіцієнт затвердіння металу виражається через коефіцієнт теплопередачі α:

$$M = \frac{\alpha}{\rho_k \cdot \gamma_M + c_M \cdot \gamma_M (T_{çàë} - T_{êð})},$$

де $\rho_к$ – питома теплота кристалізації, ккал/кг;

$\gamma_м$ – густина рідкого металу, кг/м^3;

$C_м$ – питома теплоємність рідкого металу, ккал/кг;

$T_{зал}$ – температура плавки;

$T_{кр}$ – температура кристалізації.

У середовищі VBA створити програму, яка обчислює товщину шару металу, що затвердів, для різних показників часу: від $T_{початк}$=0 до $T_{кін}$=10 хв., крок зміни температури – Δ=2хв. Прийняти α=35; $\rho_к$=23; $\gamma_м$=7000 кг/м^3; $C_м$=0,031 ккал/кг; $T_{зал}$=1400; $T_{кр}$=1147. Рішення задачі наглядно оформити на листі Excel.

11. Час зварювання металу розраховують як:

$$T = \frac{7.85 \cdot F \cdot L}{k_n \cdot I},$$

де F – площа поперечного перерізу;

L – довжина шва, см^2;

$K_п$ – коефіцієнт наплавки;

I – зварювальний струм, А.

У середовищі VBA створити програму, яка обчислює час зварювання в залежності від змінювання площі поперечного перерізу шва і зварювального струму. Значення змінюються в межах від $F_{поч}$=4,6 до $F_к$=8,2 з кроком значення I: від $I_{поч}$=2 до $I_к$=4 з кроком ΔI=0,5; L=10,5; $k_п$ =1,2. Рішення задачі наглядно оформити на листі Excel.

12. Залежність між кутом захвату заготовки, що прокачується, обтисненням та діаметром валка знаходять як:

$$\alpha = \sqrt{\frac{\Delta h}{r}},$$

де α – кут захвату, рад;

Δh – обтиснення, мм;

r – радіус балки, мм.

У середовищі VBA створити програму, яка обчислює величину обтиснення Δh для різних значень кута захвату α та радіуса r, наведених нижче:

α	$23^0 24'$	$25^0 17'$	$23^0 15'$	$20^0 18'$	$25^0 29'$
r	600	605	630	590	700

Рішення задачі наглядно оформити на листі Excel.

13. Під час прокатки листової сталі площа поверхні контакту полоси з валком (F) знаходиться за такою залежністю:

$$F = 0.5 \cdot (b_0 + b_1) \cdot \sqrt{R \cdot \Delta h},$$

де b_0 – ширина полоси до проходу крізь валки, мм;

b_1 – ширина полоси після проходу крізь валки, мм;

R – радіус валка, мм;

Δh – обтиснення полоси, мм.

У середовищі VBA створити програму, яка обчислює площу контактної поверхні полоси з валком для різних значень параметрів кліті прокатного стану, які надані нижче:

№	b_0, мм	b_1, мм	R, мм	Δh, мм
1	519	520,5	600	5
2	518	519,3	610	3
3	520	525,7	630	4
4	525	530,1	590	3
5	580	583,1	700	6

Рішення задачі наглядно оформити на листі Excel.

14. Для знаходження коефіцієнта тертя μ_y між валками стану холодної прокатки металу використовують формулу:

$$\mu_y = k_\mu \frac{0.07 \cdot 0.1 \cdot v}{3v^2 + 2v + 2},$$

де v – окружна швидкість валків, м/с;

k_μ – коефіцієнт, що враховує вплив мастила.

У середовищі VBA створити програму для обчислення значення коефіцієнта тертя для всіх видів поверхонь валків, наданих нижче:

Мастила	k_μ	n, об./хв.
Сухі чисті валки	1,55	120
Машинне масло	1,35	130
Веретенне масло	1,25	110
Вода	1	120
Емульсія	1	130
Гас	1	125
Бавовняне масло	0,9	150
Касторове масло	0,9	157
Пальмове масло	0,9	158

Окружна швидкість валка розраховується за виразом:

$V = \frac{2\pi \cdot R \cdot n}{60}$, де R – радіус валка, 0,63 м.

Рішення задачі наглядно оформити на листі Excel.

15. Для знаходження величини, на яку збільшилася ширина полоси металу, який катають Δb (уширення), використовують вираз:

$$\Delta b = 0.5 \cdot C_b \cdot C_g \cdot \left(\sqrt{R \cdot \Delta h} - 0.5 \frac{\Delta h}{m} \right) \cdot \ln\left(\frac{h_0}{h_1} \right),$$

де C_b – коефіцієнт, який враховує вплив ширини;

m – коефіцієнт тертя;

R – радіус валків;

Δh – величина обтиску полоси;

h_1 – товщина полоси до входу у валки стану;

h_0 – товщина полоси на виході з валків стану;

C_g – коефіцієнт, який враховує вплив заднього натягу (прийняти C_g =1).

У середовищі VBA створити програму, яка обчислює уширення для різних параметрів прокатки, наведених нижче:

№	C_b	m	R, мм	Δh, мм	h_0, мм	h_1, мм
1	0,6	0,086	600	5	25	20
2	0,62	0,075	605	3	20	17
3	0,65	0,07	630	4	15	11

Продовження

№	C_b	m	R, мм	Δh, мм	h_0, мм	h_1, мм
4	0,7	0,058	590	3	21	18
5	0,72	0,056	700	2	25	23
6	0,8	0,053	400	1,5	21,5	20
7	0,81	0,051	300	1,8	22	20,4
8	0,79	0,05	300	2,5	22	19,5
9	0,78	0,048	400	1,8	20	18,2

Рішення задачі наглядно оформити на листі Excel.

16. Роботу прокатки можливо визначити, коли відомі швидкість оберту валків стану, момент та час прокатки:

$$N=M\cdot n/0.975,$$

де n – число обертів валків за хвилину,

M – момент, необхідний для приводу валків.

Момент прокатки M знаходиться за виразом:

$$M = 2 \cdot P \cdot \psi \cdot \sqrt{R \cdot \Delta h},$$

де P – зусилля прокатки, т с (тон сил);

ψ – коефіцієнт плеча;

Δh – величина обтиску полоси, мм;

R – радіус валка, мм.

У середовищі VBA створити програму, яка обчислює роботу прокатки N для різних коефіцієнтів плеча ψ:0.45, 0.48, 0.3, 0.4, при цьому інші параметри дорівнюють: P=218, Δh=3, R=300, n=80. Рішення задачі наглядно оформити на листі Excel.

17. Для системи теплосупроводження прокатного цеху потрібна певна кількість одношарових та багатошарових котушок індуктивності. Індуктивність одношарової котушки знаходиться за формулою (2.1), а багатошарової – за формулою (2.2):

$$L = \frac{0.01 \cdot D \cdot \omega^2}{l/D + 0.04} \qquad (2.1);$$

$$L = \frac{0.08 \cdot D^2 \cdot \omega^2}{3D + 9l + 10t}, \qquad (2.2)$$

де L – індуктивність, мкн;

D – діаметр котушки, см;

l – довжина намотки, см;

ω – число витків;

t – товщина котушки (t=1).

Вибір будь якої котушки обумовлено співвідношенням величин ω та D. Якщо $\omega/(D\cdot 10^3)\geq 1$, обирається одношарова котушка, а інакше – багатошарова.

У середовищі VBA створити програму, яка обчислює кількість одношарових та багатошарових котушок та їх індуктивність. Параметри котушок надані нижче:

№ з/п	1	2	3	4	5	6	7	8
D	3	5	4	4	4	3.5	3.5	3.5
l	4	4.5	5	4	4.5	3.5	3.5	4
ω	300	300	600	650	300	400	200	250

Рішення задачі наглядно оформити на листі Excel.

18. Активну площу перерізу магнітопроводу для зварювального трансформатора знаходять як:

$$S = 700 \cdot \sqrt{\frac{a \cdot P}{f \cdot B \cdot \Delta}},$$

де a – коефіцієнт, що дорівнює 4,5;

P – потуга трансформатора, Ва;

f – частота живильної мережі;

B – допустима індукція, Гс;

Δ – допустима щільність струму, $а/мм^2$.

У середовищі VBA створити програму, яка обчислює активну площу перерізу магнітопроводу S для різних параметрів трансформаторів, наведених нижче:

№ з/п	P, Ва	f, гц	B, Гс	Δ, $а/мм^2$
1	10	50	7000	3,0
2	5	55	8000	4,0
3	8	57	6000	2,5
4	30	50	7000	2
5	25	55	8000	4
6	50	53	10000	2,5
7	100	52	100000	2,5

Рішення задачі наглядно оформити на листі Excel.

19. Термін твердіння піщано-глинистої смоляної суміші залежіть від температури, властивостей суміші, ступеня отвердіння, та описано виразом:

$$\tau = k_0 \cdot \beta \cdot e^{\frac{10000}{t+273}}, \text{ сек.}$$

Множник $k_0 \cdot \beta$ є функцією від заданого ступеня β твердіння та надано нижче:

β	0,85	0,9	0,95
k_0	$8{,}59 \cdot 10^{-9}$	$23{,}15 \cdot 10^{-9}$	$41{,}73 \cdot 10^{-9}$

Застосувавши середовище VBA, дослідити залежність часу охолодження від температури t, яка змінюється в діапазоні від $t_{поч}$=600 ^{0}C до $t_{кін}$=1000 ^{0}C з кроком Δt=100^{0}C. Рішення задачі наглядно оформити на листі Excel.

20. Для контролю нагріву зливків у колодязях потрібно знати їх середню початкову температуру. Приблизна залежність для розрахунку перепаду температур між центром та поверхнею зливка має вигляд:

$$\Delta t = \frac{\alpha_{îõ} \cdot (t_{ïîâ} - t_{ñåð}) rR}{2\lambda},$$

де $\alpha_{охл}$ – коефіцієнт тепловіддачі від зливка до повітря:

$t_{пов}$ – температура поверхні зливка, ^{0}C;

$t_{сер}$ – температура середовища, ^{0}C;

λ – теплопровідність зливка, кВт/(м^{0}C);

R – розрахований радіус зливка, м.

Температуру центру $t_ц$ зливка й середню температуру $t_{сер}$ знаходять відповідно за виразами: $t_ц = t_{пов} + \Delta t$ та $t_{сер} = t_{пов} + 0{,}5 \cdot \Delta t$.

У середовищі VBA створити програму, яка розраховує залежність $t_{сер}$ та $t_ц$ від $t_{пов}$ та $\alpha_{охл}$ для зливків з радіусом R=0.3 м, α=0.025, $t_{сер}$=20^{0}C. Температура поверхні і коефіцієнт тепловіддачі від зливка до повітря наведено нижче:

$t_{сер}$	1120	730	610
$\alpha_{охл}$	0,184	0,049	0,065

Рішення задачі наглядно оформити на листі Excel.

21. Температура зливка у секції колодязя в процесі нагріву знаходиться як:

$$t_i = t_{i-1} + \frac{\Delta\tau}{T}(t_{ß,i-1} - t_{i-1}),$$

де t_{i-1}, t_k – температура зливка у попередній та наступний моменти часу, ^{0}C;

$\Delta\tau$ – крок інтегрування, хв.;

T – стала часу, хв.;

$t_{Я,\,i-1}$ – температура секції у попередній момент часу, ^{0}C.

Час, який залишився до кінця нагріву, розраховується за виразом:

$$\grave{O}_{\hat{I}\times\ ,i} = T \cdot \ln \frac{t_{ß\,,i} - t_i}{t_{ß\,,^3} - t_{ÇÀÄ}},$$

де $t_{ЗАД}$ – температура нагріву зливка, що задана, ^{0}C.

Під час розрахунку $T_{Оч,\,i}$ повинна виконуватися умова:

$$t_{Я,\,i} > t_i \quad \text{і} \quad t_{Я,\,i} > t_{ЗАД}.$$

У середовищі VBA створити програму, яка розраховує t_i та $T_{Оч,\,i}$ по відомих температурах секції у задані моменти часу та T=200 хв., $\Delta\tau$=20хв., $t_{ЗАД}$=1150^{0}C, t_0=500^{0}C, $t_{Я,0}$=800^{0}C, $t_{Я,1}$=950^{0}C, $t_{Я,2}$=1000^{0}C, $t_{Я,3}$=1100^{0}C, $t_{Я,\,4}$=1180^{0}C. Рішення задачі наглядно оформити на листі Excel.

22. Дано ряд розподілу випадкової величини X:

X_i	0	1	2	3	4	5
P_i	0,5	0,3	0,15	0,025	0,015	0,010

У середовищі VBA створити програму, яка розраховує величини Mx, Dx та Gx за виразами:

$$Mx = \sum_{i=1}^{6} X \cdot i \cdot P_i, \qquad Dx = \sum_{i=1}^{6} (X_i - Mx)^2 \cdot P_i, \qquad Gx = \sqrt{Dx}.$$

Рішення задачі наглядно оформити на листі Excel.

23. Вміст вуглецю в чавуні доменної плавки можна обчислити за наближеною формулою 2.3.

$$C=1{,}3+2{,}57\cdot 10^{3\cdot}t, \qquad (2.3)$$

де t – температура чавуну на випуску (t=1500^{0}C).

Більш точніше вміст вуглецю можна знайти за формулою 2.4.

$$C'=4{,}3-0{,}3\cdot Mn-0.27\cdot Si-0.32\cdot P-0.032\cdot S, \qquad (2.4)$$

де Mn, Si, P, S – процентний вміст марганцю, кременю, фосфору та сірки в чавуні.

У середовищі VBA створити програму, яка дає можливість розрахувати та порівняти вміст вуглецю в чавуні, обчислений за формулами 1 та 2. Обчислити $\Delta C = C' - C$. Вміст елементів в чавуні Si=0,64; P=0,03; S=0,028. Вміст марганцю змінюється в межах від 0,15 до 0,20 з кроком 0,01. Рішення задачі наглядно оформити на листі Excel.

24. Площа перерізу живильників літникової системи розраховується як:

$$F = \frac{G}{\mu \cdot \tau \cdot 0.31 \cdot \sqrt{H_{cp}}},$$

де G – сума мас виливки літників та прибутків, що приходяться на цю виливку, кг;

μ – загальний коефіцієнт витрати в літниковій системі;

τ – тривалість заливки, хв.;

$H_{ср}$ – середній розрахований напір.

У середовищі VBA створити програму, яка дає можливість розрахувати площу перерізу живильників літникової системи, якщо час заливки змінюється від $\tau_п$=2 хв до $\tau_к$=10 хв з кроком Δτ=2 хв, при μ=0,5; $H_{ср}$=1,5; G=2,5 кг. Рішення задачі наглядно оформити на листі Excel.

25. У середовищі VBA створити програму, яка дає можливість розрахувати коефіцієнт тертя f, що забезпечує нерухомість вантажу на горизонтальній платформі, який обертається, коли число обертів за хвилину дорівнює 8, 10, 14, 15, 20, 30, 50, 100.

Розрахунки виконати для випадків, коли вантаж знаходиться на відстані від осі обертання R 20, 50 та 75 см. Для розрахунків використовувати співвідношення:

$$F = 4 \cdot \pi^2 \cdot n^2 \cdot \frac{R}{60},$$

де n – число обертів платформи, об./хв.;

R – відстань вантажу від осі обертання.

Рішення задачі наглядно оформити на листі Excel.

26. Граничний стан процесу поширення при автоматичному зварюванні в стик описується рівнянням:

$$T(y,\tau) = \frac{q}{v \cdot \delta \cdot \sqrt{4 \cdot \pi \cdot \lambda \cdot c \cdot \gamma \cdot \tau}} \cdot e^{\left(-\frac{y^2}{4 \cdot a \cdot \tau} - b \cdot \tau\right)} + T_0,$$

де T(y, τ) – температура пластини на відстані ***y*** від осі шва в момент часу ***τ***, К;

q – ефективна потужність дуги, Вт;

v – швидкість зварки, см/с;

δ – товщина пластини, см;

λ – коефіцієнт теплопровідності, Дж/(см·с·К);

c – середня теплоємність, Дж/(г·К);

γ – щільність, г/см^3;

a – коефіцієнт температуропровідності, см^3/с;

b – коефіцієнт температуровіддачі, 1/с;

T_0 – температура металу перед зваркою, К.

У середовищі VBA створити програму, яка дає можливість розрахувати залежність зміни температури пластини від відстані до осі зварного шва. Значення y змінюються в межах від y_0=0 до $y_к$=5 з кроком Δy=1; T_0=20 К; q =400 Вт; v=2,1 см/с; δ=35 см; λ=0,418; c=0,669; γ=7,8; a=0,08; b=6,4·10^4; $\boldsymbol{\tau}$=1. Рішення задачі наглядно оформити на листі Excel.

27. Температура граничного стану процесу наплавки дугою валика на масивний виріб в координатах xyz задається співвідношенням:

$$T = \frac{g}{2 \cdot \pi \cdot \lambda \cdot R} \cdot e^{\left(-\frac{v}{2a} \cdot (x+R)\right)},$$

де R=x^2+y^2+z^2;

a – коефіцієнт температуропровідності, см^3/с;

λ – коефіцієнт теплопровідності, кал/(см·с· ^{0}C).

У середовищі VBA створити програму, яка розраховує температуру T при наступних вихідних даних: g=1000 кал/с; v=0,1 см/с; a=1,4·10^{-3}см^2/с; λ=1,4·10^{-3} кал/(см·с· ^{0}C); z=2,0; **x** змінюється в межах від $\mathbf{x}_{поч}$=1,0 до $\mathbf{x}_{кін}$=4,0 з кроком Δ**x**=0,5; **y** змінюється в межах від $\mathbf{y}_{поч}$=0,5 до $\mathbf{y}_{кін}$=2,0 з кроком Δ**y**=0,5.

Рішення задачі наглядно оформити на листі Excel.

28. Пальник автогенного різака рухається в площині XOУ за законом:

$$X = \frac{3 \cdot t}{1+t^3}, м \; ; \qquad Y = \frac{3 \cdot t^2}{1+t^3}, м,$$

де X,У – координати площини, м;

t – час, секунди.

У середовищі VBA створити програму, яка визначає швидкість та прискорення руху різака у моменти часу: t_1=1 с; t_2=3 с; t_3=5 с; t_4=7 с; t_5=12 с. Рішення задачі наглядно оформити на листі Excel.

Для розв'язання задачі використати наступну послідовність обчислень:

Складова швидкості різака по осі X: $Vx = X'' = \frac{3 - 6t^3}{\left(1+t^3\right)^2}$.

Складова швидкості різака по осі У: $Vy = Y'' = \frac{6t - 3t^4}{\left(1 + t^3\right)^2}$.

Сумарна швидкість: $V = \sqrt{V_x^2 + V_y^2}$

Складова прискорення різака по осі Х: $Wx = X''' = \frac{18t^2 - 18t^5 - 36t^3}{\left(1 + t^3\right)^4}$.

Складова прискорення різака по осі У: $Wy = Y'' = \frac{6t^3 - 36t^6 - 36t^3 + 6}{\left(1 + t^3\right)^4}$.

Сумарне прискорення: $W = \sqrt{W_x^2 + W_y^2}$.

29. У середовищі VBA створити програму, яка дозволяє визначити залежність коефіцієнта корисної дії редуктора μ від поточного значення потужності N для потужності під час підйому номінального вантажу Nc, яка дорівнює відповідно 2,5; 4,0 та 7,5 кВт. При цьому, N змінюється від 5,0 до 6,0 кВт з кроком 0,1. Обчислення проводити за виразом:

$$\eta = \frac{\eta_n}{1.05 + \frac{0.05 \cdot Nc}{N}} ,$$

де η_n – коефіцієнта корисної дії поліспаста, η_n =0,67.

Рішення задачі наглядно оформити на листі Excel.

30. Секундний об'єм води розраховується як:

$$V = \frac{\pi \cdot R^4}{8 \cdot \eta \cdot l} \Delta P \text{ , м}^3\text{/с ,}$$

де R – радіус циліндричної труби, м;

η – динамічна в'язкість рідини, Па·с;

l – довжина частини труби, на якій можливо прийняти ΔР= const, м;

ΔР – падіння тиску рідини на частині труби довжиною *l*, Па.

У середовищі VBA створити програму, яка дозволяє розрахувати значення V, коли радіус труби змінюється від $R_{ахч}$=0,1 м; $R_{кін}$=0,3 м; ΔR=0,025 м; для l_1= 2 м; l_2=2,5 м; l_3=3 м. Вихідні дані: η=0,0018 Па·с; ΔР=0,02 Па. Рішення задачі наглядно оформити на листі Excel.

2.4 Запитання для самоконтролю

1. Що таке об'єктно-орієнтоване програмування?
2. Основні концепції об'єктно-орієнтоване програмування.
3. Що таке VBA?
4. Основні складові середовища VBA.
5. Елементи управління VBA та їх властивості.
6. Для чого застосовують написи?
7. Що таке змінна у VBA?
8. Перелічте правила утворення імен змінних.
9. Назвіть типи змінних у VBA.
10. Що таке об'єкт, подія, процедура у VBA?
11. Які мовні конструкції для реалізації розгалужень має VBA?
12. Назвіть різновиди умовного оператора *IF*.
13. Призначення стандартної функції *Val?*
14. Перелічте різновиди стандартних функцій Visual Basic.
15. Яка властивість елемента *Label* дозволяє змінити текст на цьому об'єкті?
16. Назвіть властивість елемента управління *CommandButton,* що дозволяє змінити текст (надпис) на цьому об'єкті у VBA.
17. Поясніть, як запустити проект на виконання у VBA.
18. Що таке «змінна циклу», «тіло циклу», «крок циклу»?
19. Який синтаксис має оператор циклу *For... Next* і як він працює у VBA?
20. Яке призначення має директива компілятора Option Base 0*?*
21. Як вивести на екран вікно, що містить повідомлення і поле введення строкових даних?
22. Пріоритети операторів VBA.

3 СЕРЕДОВИЩЕ BORLAND C++ BUILDER

Borland C ++ Builder поєднує простоту середовища швидкої розробки додатків, або RAD-середовища (Rapid Application Development), з потужністю і продуктивністю мови C ++, сумісної зі стандартом ANSI.

Цей стандарт дозволяє створювати і компілювати програми в різних системах розробки, для різних операційних систем і наборів процесорних інструкцій. Основна частина роботи зі створення додатків виконується в інтегрованому середовищі розробки (Integrated Development Environment – IDE) C ++ Builder.

Мова C ++ є ядром середовища C ++ Builder, яке забезпечує дуже високий ступінь підтримки цієї стандартизованої мови програмування.

3.1 Головні складові частини середовища C++ Builder

C ++ Builder являє собою додаток, головне вікно якого (рис. 3.1) містить:

1. Головне меню.
2. Панелі інструментів.
3. Вікно *Дизайнер Форм (Form1)* – це порожня форма, яка автоматично з'являється на екрані при створенні проекту. Тут розташовуються компоненти середовища. Щоб сформувати призначений для користувача інтерфейс, досить додати в форму відповідні візуальні компоненти, а потім вказати їх розташування і розмір.
4. Палітру компонентів (*Component Palette*). Вона розташовується відразу під рядком головного меню і містить всі компоненти VCL (Visual Component Library). Ці компоненти згруповані в окремих вкладках і розбиті за категоріями, назви яких вказані у верхній частині вкладок. Властивості компонентів можна модифікувати за допомогою вікна Object Inspector. Крім того, є можливість змінювати розміри або розташування.
5. Вікно *Інспектор об'єктів (Object Inspector)* – це вікно, що дозволяє побачити основні властивості і події об'єкта, поміщеного у форму.

Властивості є атрибутами компонента, що визначають його зовнішній вигляд і поведінку. Багато властивостей компонента в колонці властивостей мають значення, яке встановлюється за замовчуванням (наприклад, висота

кнопок). Властивості компонента відображаються на сторінці властивостей (Properties).

Сторінка подій *(Events)* інспектора об'єктів показує список подій, розпізнаваних компонентом. Кожен компонент має свій власний набір обробників подій.

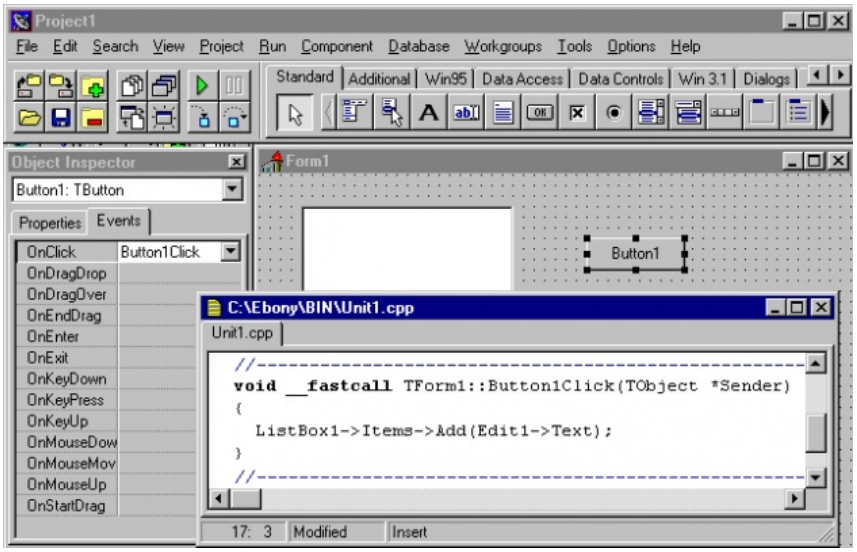


Рисунок 3.1 – Головне вікно середовища C++ Builder

– вікно редактора коду – тут знаходиться програмний модуль (за умовчанням ім'я Unit1.cpp).

3.2 Компоненти C ++ Builder

Компоненти розділяються на видимі – візуальні і невидимі – невізуальні (рис.3.2). Візуальні компоненти з'являються під час виконання точно також, як і під час проектування. Прикладами є кнопки и редаговані поля. Невізуальні компоненти з'являються під час проектування як піктограмі на формі. Вони непомітні під час виконання, але мають певну функціональність.

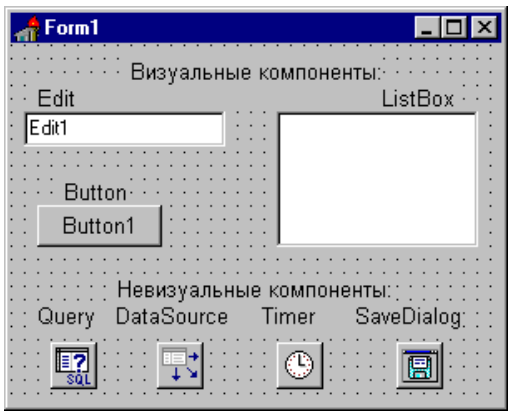


Рисунок 3.2 – Компоненти на формі

Для додавання компонента у форму можна вибрати мишею потрібний компонент в палітрі і клацнути лівою клавішею миші в потрібному місці проектованої форми. Компонент з'явиться на формі, і далі його можна переміщати, змінювати розміри та інші характеристики.

Кожен компонент C ++ Builder має три різновиди характеристик: властивості, події, методи.

Якщо вибрати компонент з палітри і додати його до форми, інспектор об'єктів автоматично покаже властивості та події, які можуть бути використані з цим компонентом.

3.3 Проект Builder

Усі програми користувача в середовищі Builder оформлюються у вигляді *проектів*. Розробка будь-якого проекту починається зі збереження порожнього проекту в новій папці: пункт меню **File → Save Project As**:

- програмний модуль: за умовчанням ім'я **Unit1.cpp** → зберегти;
- модуль проекту: за умовчанням ім'я **Project1.bpr** → зберегти.

Проміжні збереження:

пункт меню **File→Save All** – зберігаються усі початкові файли під поточними іменами.

Папку, що містить файли проекту, можна переміщати, перейменовувати, переносити на інші комп'ютери, але файли, що відносяться безпосередньо до проекту, перейменовувати не можна.

Результатом роботи проекту є здійснимий файл (додаток).

Склад і структура проекту

Середовище Builder зв'язує з кожним своїм застосуванням наступні файли:

Unit1.cpp – модуль форми (програмний модуль). Якщо з'являться нові форми, то для кожної з них буде побудований свій програмний модуль: Unit2.cpp, Unit3.cpp і так далі. Переглянути цей файл можливо за допомогою пункту меню **File→Open** або **View→ Units.**

Unit1.h – модуль з оголошеннями компонентів, які розташовані в цій формі. Для кожної нової форми буде побудований свій модуль: Unit2.h, Unit3.h,

і так далі. Переглянути цей файл можливо за допомогою контекстного меню Редактора кода → **Open Source/Hear File.**

Unit1.dfm – файл опису форми і усіх розташованих в ній компонентів. Переглянути цей файл можливо за допомогою контекстного меню Форми → **View As Text**

Project1.cpp – файл проекту. Це головна програма, що управляє проектом. Переглянути файл можливо за допомогою пункта меню **Project→ View Source**.

Project1.bpr – файл, що відповідає за проведення процесу збирання і компіляції проекту.

Project1.exe – здійснимий файл (додаток), готова до виконання програма, яка може функціонувати під управлінням операційної системи Windows.

Project1.res – файл ресурсів проекту, є двійковий файл. У ньому зберігаються різні значки, графічні зображення, використовувані в проекті.

Якщо переглянути папку, в якій знаходиться весь проект, то виявимо ще й інші файли, наприклад файли тимчасового зберігання ~ cpp, ~ dfm, ~ obj.

3.4 Елементи мови С++

Під елементами мови розуміють базові конструкції, які використовуються при написанні програм. До них відносяться: алфавіт, правила запису констант і ідентифікаторів, основні типи даних і дії над ними.

Символи мови

1. Прописні і рядкові букви латинського алфавіту A, B, C, .X, Y, Z;
   a, b, c,.x, y, z; символ підкреслення _ .
2. Прописні і рядкові букви російського алфавіту

Увага. *Треба пам'ятати, що однакові прописні і рядкові букви вважаються різними символами.*

3. Арабські цифри 0, 1, 2, …,.9.
4. Спеціальні символи . ; ( ) < > / * = – % та інші.
5. Управляючі символи, які використовуються у функціях введення-виводу :

- \n – перехід на новий рядок;
- \t – горизонтальна табуляція;

- \o – нульовий символ;
- \v – вертикальна табуляція та інші.

Константи

У мові С++ розрізняють чотири види констант:

Цілі константи:

– *десяткові*: наприклад 16; 240;

– *вісімкові:* в якості цифр використовуються символи 0, 1, 2, 3, 4, 5, 6, 7. Вісімкові константи завжди починаються з нуля, наприклад:

01, 065, 0777;

– *шістнадцяткові:* в якості цифр використовуються 16 символів – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Шістнадцяткові константи завжди починаються з пари символів 0x. Наприклад:

0x10, 0xFFF, 0x1F1A

Дійсні константи (з плаваючою точкою)

Наприклад: 3.45; 1.5E-2; -1.85E12; -.56.

Символьні константи

Це символ, поміщений в апострофи. У таблиці кодів ASCII кожному символу ставиться у відповідність ціле позитивне число (код), тому *значенням символьної константи є числовий код символу.*

Приклади символьних констант : 'Q'; 'a'; ''.

Строкові константи – це послідовність символів, поміщена в лапки. Наприклад: "Borland C++".

У кінці строкової константи завжди стоїть ознака кінця рядка '\0', яку формує компілятор: "Borland C++'\0'".

Коментарі

Коментар – це частина програми, яка ігнорується компілятором і служить для зручності читання вихідного тесту. В процесі компіляції коментар замінюється пропуском.

Якщо текст коментарів займає один рядок, то використовується конструкція //*коментар*… Якщо текст коментарів займає більше за один рядок, то використовується конструкція /* */.

Типи даних

Особливістю мови С++ являється відсутність принципу умовчання. Тому типи усіх змінних мають бути оголошені до їх використання.

Базові типи даних:

int цілий тип;

float тип з плаваючою точкою;

double з плаваючою точкою подвійної точності;

char символьний тип;

void порожній тип, не маючий значення.

Це основні типи даних (табл.3.1). На основі цих типів будуються інші типи за допомогою модифікаторів:

short короткий;

long довгий;

unsigned беззнаковий;

bool логічний тип.

Таблиця 3.1 – Основні типи даних С++

Тип	Розмір, (байт)	Значення
bool	1	true, false
int	2	-32768 – 32768
short int	2	-32768 – 32767
long int	4	-2147483648 – 2147483648
unsigned int	2	0 – 65535
float	4	-1,2e-38 – 34e38
double	8	2,2e-308 – 1.8e308
char	1	256 значень символів

У наведеній таблиці розмір в байтах і інтервал зміни можуть варіюватися в залежності від компілятора, процесора і операційної системи (середовища.)

Оголошення та ініціалізацію змінних розглянемо на прикладах.

1) **int** a = 24, i = 5; //змінним цілого типу **a, i** привласнюються значення
2) **char** c='c'; //змінної символьного типу **c** привласнюється значення

Змінна **c** містить один символ **c** (точніше, його код по таблиці кодування), тому типи **char** і **int** взаємозамінні.

3) **String** a, b; // оголошуються змінні строкового типу **a** і **b**

a = "Програмування";

b = "на C++".

Змінна будь-якого типу може бути оголошена як та, що не змінюється. При цьому додається зарезервоване слово **const**.

Наприклад: **const double** A=2.12E-2.

Увага. Вибирайте тип змінних з урахуванням діапазону і необхідної точності представлення даних. Давайте змінним імена, що відображають їх значення.

Масиви

Оператор опису масиву задає його тип, ім'я та розмірність (кількість елементів). Всі інструкції виділення пам'яті формує компілятор до виконання програми, тому розмірність масиву може бути тільки константою або константним виразом. При описі масив можна ініціалізувати (привласнити його елементам початкові значення).

Елементи масиву нумеруються з нуля, тому максимальний індекс (номер) елемента завжди на одиницю менше розмірності. Автоматичний контроль виходу індексу за межі масиву не виконується, тому програміст повинен стежити за цим самостійно.

Оголошення масивів

1) **int** a[30]; //масив з 30 елементів цілого типу, нумерація з нуля

Це елементи: a[0], a[1], ...a[29]. Масив займає в пам'яті 2*30=60 байт.

2) **double** b[2][3]; /*двовимірний масив з 6 елементів з плаваючою точкою подвійної точності */.

Це елементи: b[0][0], b[0][1], b[0][2], b[1][0], b[1][1], b[1]][2]].

Масив зберігається по рядках в безперервній області пам'яті. Для доступу до окремого елемента масиву застосовується конструкція виду b[i][j], де i (номер рядка), а j (номер стовпчика) – вираження цілого типу.

Увага. Перший індекс завжди сприймається як номер рядка, другий – як номер стовпчика, незалежно від імені.

Ініціалізація масивів

1) Одновимірний масив

int m[2]={1,8};

2) Двовимірний масив $A = \begin{pmatrix} 2 & 1 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

int a[2][3]={{2, 1, 3}, {4, 5, 6}};

можливі варіанти **int** a[2][3]={2, 1, 3, 4, 5, 6};

int a[ ][3]={2, 1, 3, 4, 5, 6};

3) Масив символів

char str[3]={'a ',' b ',' c'}; //одновимірний масив символів

Арифметичні операції

+	складання	**/**	цілочисельне ділення
–	віднімання	**%**	залишок від ділення
*****	множення		унарні операції **+** і **-**

Наприклад:

операція	**результат**
11/3	3
11./3.	3.666….
11%3	2
-x*y	(–x)*y, оскільки пріоритет унарних операцій вищий, ніж у бінарних.

Таблиця 3.1 – Математичні функції мови С++

Функція	**Опис**
fabs	модуль (абсолютна величина)
abs	модуль (для цілих чисел)
sin	синус
cos	косинус
tan	тангенс
exp	експонента, тобто піднесення числа **e** (основа натурального логарифма) до зазначеного ступеня
log	натуральний логарифм
log10	десятковий логарифм
sqrt	корінь квадратний
pow(x, y)	піднесення до степеня x^y
pow10(x)	піднесення до степеня 10^x
asin	арксинус
acos	арккосинус

Продовження таблиці 3.1

Функція	Опис
atan	арктангенс
fmode(x, y)	залишок від ділення x на y
M_PI	Число π

Усі функції, окрім **abs,** повертають значення типу **double,** типи аргументів повинні бути теж **double**.

Приклади запису арифметичних виразів на мові С++ :

$$\frac{e^{2x}+\sin(x-y)^2}{\sqrt{xy}-\ln\frac{x}{2}} \quad \rightarrow \quad \text{(exp(2*x)+sin((x-y)*(x-y)))/(sqrt(x*y) - log(x/2.))}$$

$$\sin^2 x \quad \rightarrow \quad \text{pow(sin(x), 2)}$$

$$\sqrt[3]{x} \quad \rightarrow \quad \text{pow(x, 1./3.)}$$

Функції в С++

Усі програми на С++ будуються з функцій. Функція – це самостійна одиниця програми, яка створена для вирішення конкретного завдання. Функція в мові С++ відіграє ту саму роль, що і підпрограми або процедури в інших мовах. Кожна функція має ім'я і список аргументів, інколи може бути відсутнім (табл.3.1). Одна частина функцій знаходиться у бібліотеках. Інша − створюється самим користувачем (функції користувача).

Кожна функція перед використанням має бути оголошена. Оголошення бібліотечних функцій знаходяться в заголовних файлах, які підключаються до програми за допомогою директиви **#include** <ім'я файлу.h>.

Оголошення математичних функцій підключаються за допомогою директиви **#include <math.h>.**

3.5 Консольний режим. Можливості введення-виводу С++

Консольний режим – це програма на мові С++ в середовищі Builder, яка запускається без графічного інтерфейсу в консольному вікні.

Введення, що йде з клавіатури:

cin >>

cin – стандартное ім'я потоку введення.

Наприклад:

а) **cin** >> a; // дані вводяться з клавіатури в змінну **a**

б) **cin** >> i >> j >> s; //дані вводяться в три змінні i, j, s

Виведення, що йде на екран:

cout <<

cout – стандартне ім'я потоку виводу.

Наприклад:

а) **cout** <<b; // значення змінної **b** виводиться на екран

Оголошення функцій *cin, cout* підключаються до проекту за допомогою директиви **#include <iostream.h>**.

Створення консольного додатка. Лінійний обчислювальний процес

Працюючи в консольному режимі, завжди має бути функція з ім'ям **main** (головна), саме з неї починається виконання програми, в якому б місці вона не знаходилася.

Приклад. Розробка проекту для обчислення значення функції ***y*** при різних значеннях ***x,*** які будуть вводитися з клавіатури.

$$y = a \cdot e^{x} + \ln c\,,$$

де $c = \sqrt{b + s \cdot \sin\left(\frac{\pi}{4}\right)}$; a = 3,112; b = 5,85; s = 9, 48**.**

Порядок дій :

1. Створити новий проект за допомогою пункту меню **File→New**.
2. У вікні, що відкрилося, вибрати **Console Wizard** → OK; активізувати перемикач C++, потім **Console Application** → OK;

На екрані з'явиться вікно **Unit1.cpp** і заготівля для введення функції:

```
int main(int argc, char* argv[])
{
    return 0;
}
```

3. Зберегти проект за допомогою пункту меню **File → Save Project As:** створити нову папку, зберегти 2 модулі:
   - ім'я програмного модуля за умовчанням **Unit1.cpp;**
   - ім'я модуля проекту **Project1.bpr**.

4. Внести зміни в заготівлю, програма на С++ в консольному режимі має вигляд:

```
#include <math.h>                    //для математичних функцій
#include <iostream.h>                //для функцій cin, cout
#include <conio.h>                   //для функцій getch(), clrscr ()

   //---------------------------------------------------------------------

main(){                                        //це головна функція
   double a=3.112, b=5.85, s=9.48;             //ініціалізація змінних
   double  x, y, c;                            //оголошення змінних
   clrscr ();                                  //функція очищення екрану
   cout<<"input x"<<endl;      //запрошення до вводу і спуск на новий рядок
   cin>>x;                                     //введення з клавіатури значення x
   c= sqrt(b+s*sin(M_PI/4));
   y=a*exp(x)+log(c);
   cout<<"y="<<y;
   getch();
}
```

Функція getch() чекає введення з клавіатури будь-якого символу, роблячи при цьому затримку екрану виводу.

5. Запустити проект на виконання RUN [F9]:
   при x=5,5 ми одержимо результат y=762,421.
6. Зберегти відлагоджену програму File→ Save All.

Увага. Введення з клавіатури випереджайте запрошенням, а значення, що виводяться – поясненнями. Дані при введенні розділяйте пробілами, символами переведення рядка або табуляції.

Стадії проходження програми

На рисунку 3.3 представлена схема стадій виконання (проходження) програми.

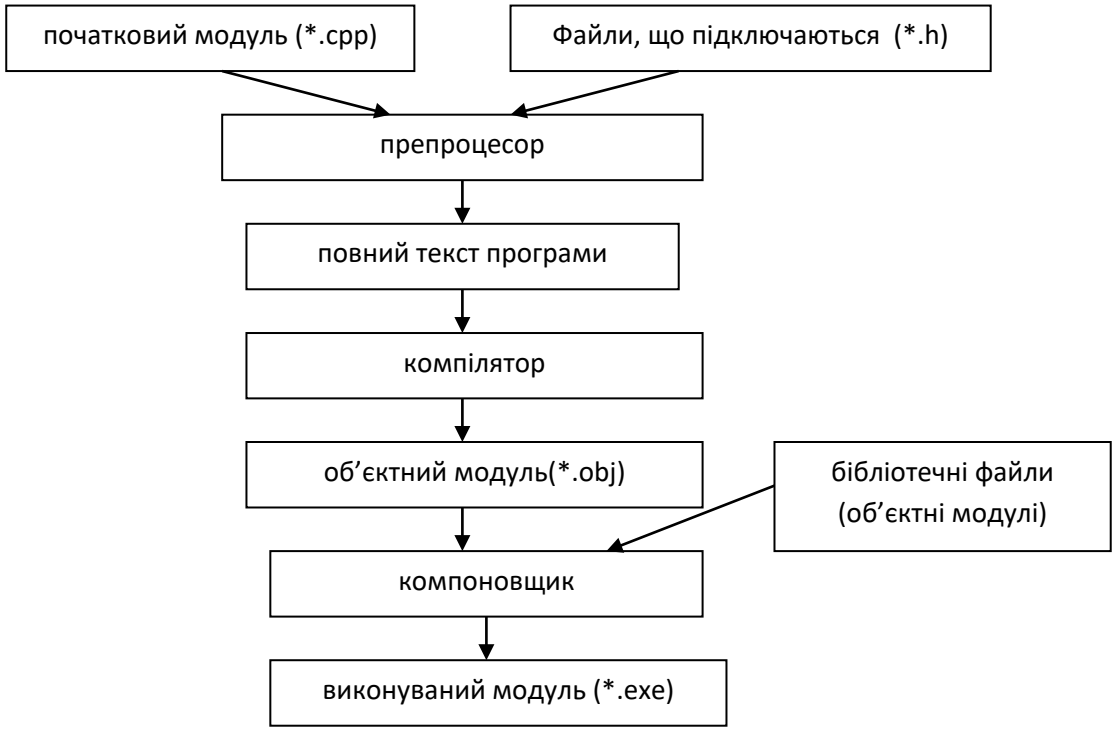


Рисунок 3.3 – Стадії проходження програми

1. *Початковий модуль* (*.cpp) – це програма на алгоритмічній мові С++. Він передається препроцесору, який виконує директиви, що знаходяться в тексті. У текст програми включається вміст вказаних файлів.

#include <ім'я файлу.h> – пошук файлу здійснюється у бібліотеці ОС (бібліотечні файли).

#include "ім'я файлу.h" – пошук файлу здійснюється в поточному каталозі (функції, створені користувачем).

Повний текст програми передається на вхід компілятора.

2. Компілятор виявляє синтаксичні помилки і у разі їх відсутності будує *об'єктний модуль* (*.obj) (програму в двійкових кодах без синтаксичних помилок).

3. Компоновщик, підключаючи до об'єктного модуля інші об'єктні модулі (бібліотечні файли), формує *виконуваний модуль*. Виконуваний модуль має розширення *.exe – це готова до виконання програма.

4 WINDOWS-ДОДАТКИ У ГРАФІЧНОМУ СЕРЕДОВИЩІ

При всьому різноманітті алгоритмів розв'язання задач в них можна виділити три основні види обчислювальних процесів: лінійний, розгалужений та циклічний.

4.1 Лінійний обчислювальний процес

Лінійним називається такий обчислювальний процес, при якому всі етапи рішення задачі виконуються в природному порядку проходження записи цих етапів.

Функції приведення типів

StrToInt(рядок)	перетворення рядка в ціле число
StrToFloat(рядок)	перетворення рядка в число з плаваючою точкою
IntToStr(ціле число)	перетворення цілого10-го числа в строковий тип
FloatToStr(число)	перетворення значення з плаваючою точкою в строкове представлення
IntToHex(ціле число)	перетворення цілого 10-го числа в 16-не число

Операції з рядками

1. Злиття рядків. Вивід здійснюється в об'єкт Label1:

```
Label1 –>Caption="Мова" "C++"
```

Результат => Мова C++

2. Злиття рядка і числовим виразом :

```
Label2 –>Caption =" Результат=" + IntToStr(4*6);
```

Результат => 24

Введення даних з клавіатури

У графічному режимі Builder введення даних з клавіатури здійснюється за допомогою функції **InputBox,** яка підключається до проекту за допомогою директиви: **#include <Stdlib.h>**. Функція повертає значення типу **String.**

Загальний вигляд функції:

InputBox ("повідомлення", "підказка", "Default")

Default – строковий вираз, який використовується за замовчуванням, якщо користувач не введе рядок. У загальному випадку може бути пропуск.

Приклад. Створювання проекту для перекладу кількості дюймів, що вводяться з клавіатури, в сантиметри; 1дюйм = 2,54см.

Порядок дій:

– створити проект, використовуючи пункт меню **File → New Application;**

– зберегти проект за пунктом меню **File → Save Project As;**

– розмістити об'єкти, надати їм необхідні властивості у *вікні властивостей.*

На сторінці **Standard** палітри компонентів обрати:

- об'єкт **Label1** – для виведення кількості дюймів;
- об'єкт **Label2** – для виведення кількості сантиметрів;
- об'єкт **Button1** – для запуску проекту, задати властивість *Caption* → ОБЧИСЛИТИ;
- об'єкт **Button2** – для очищення полів виводу, обрати властивість *Caption* → ОЧИСТИТИ.

Обробник події *Click* кнопки *ОБЧИСЛИТИ* має вигляд:

```
#include <Stdlib.h>
//=========================================
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float a, inch;
    inch=StrToFloat(InputBox("input", "kd"," "));        //введення кількості дюймів
    a=inch*2.54;                                          //обчислення кількісті сантиметрів
    Label1 ->Caption="кількість дюймів ="+FloatToStr(inch);
    Label2 ->Caption="кількість см ="+FloatToStr(a);
}
```

Обробник події *Click* кнопці *ОЧИСТИТИ* має вигляд:

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  Label1 ->Caption="";
  Label2 ->Caption="";
}
```

Запустити програму на компіляцію і виконання **[F9] (Run).** Результат роботи коду надано на рисунку 4.1:

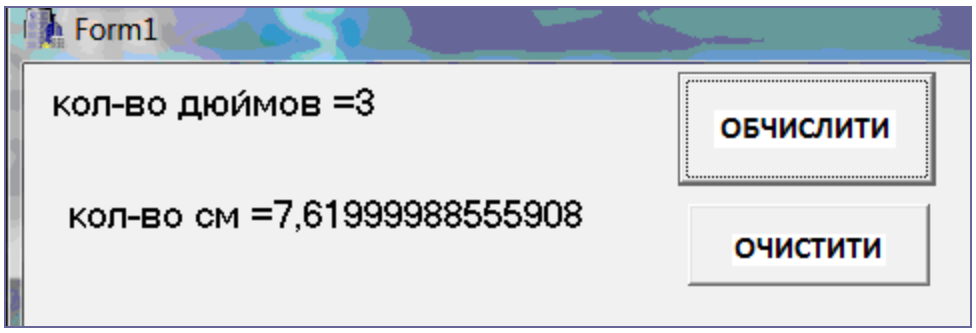


Рисунок 4.1 – Результат виконання коду

Увага. При використанні стандартних функцій підключайте до програми відповідні заголовні файли за допомогою директиви *#include*.

Перетворення типів

Якщо у програмі використовуються змінні різних типів, то компілятор автоматично здійснює їх перетворення.

Правила перетворень

1. У операторові привласнення тип значення правої частини завжди перетвориться в тип значення лівої частини.

Наприклад:

int i = 3.14; // 3.14 перетвориться до цілого типу, в результаті i =3.

2. У арифметичному виразі нижчий тип завжди перетвориться до вищого, наприклад:

float → double, int → long int.

3. Будь-який вираз може бути приведений до бажаного типу за допомогою конструкції: **(тип) вираз.**

Наприклад: int x=5;

(float) x/2;

Значенням виразу буде 2.5

Операції збільшення і зменшення на одиницю

У програмуванні часто зустрічаються ситуації, коли треба збільшити або зменшити значення деякої змінної на одиницю. Для цього зазвичай використовується оператор присвоювання виду: S= S+1. У мові C++ для цих цілей існують спеціальні операції:

++ збільшити на одиницю;

– – зменшити на одиницю.

Для операцій збільшення і зменшення на одиницю необхідний один операнд (унарні операції). Ці операції називаються інкремент (++) і декремент (– –).

Якщо збільшення і зменшення використовується у виразі, то при цьому суттєво, з якого боку від імені стоїть знак ++ або – –. Наприклад:

++ c; //значення ***c*** збільшується на одиницю, а потім використовується;

c++; //значення ***c*** використовується, а потім збільшується на одиницю.

Приклад. Що буде виведене на екран в результаті роботи фрагменту коду?

```
c=5;
x=c++;              //x=5, c=6
cout<<x<<c;
```

результат 5 6

4.2 Обчислювальний процес, що розгалужується

Розгалуженим називається такий обчислювальний процес, в якому вибір напрямку обробки інформації залежить від вихідних або проміжних даних (від результатів перевірки виконання будь-якого логічного умови).

Операції відносин

Операції відносин використовуються для порівняння. Повний список операцій відносин в мові С ++ наступний:

= =	рівно	**!=**	нерівно
>	більше	**>=**	більше або рівно
<	менше	**<=**	менше або рівно

Якщо дві змінні порівнюються операцією відносин, то результат завжди буде *істина* або *неправда*. Якщо результат *істина (true)*, то формується значення, відмінне від нуля, найчастіше одиниця. Якщо результат *неправда (false)* – формується значення, що дорівнює нулю.

Логічні операції

Логічні операції в С++ відповідають класичним логічним операціям, а їх результат відповідає таблицям, які прийнято називати **таблицями істинності**. Таблиця істинності наведена нижче:

x	**y**	**x && y**	**x || y**	**!x**
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

1 – істина; 0 – неправда.

&& логічне *і* (AND)

| | логічне *або* (OR)

! логічне *ні* (NOT)

Пріоритет логічних операцій нижче пріоритету операцій відносин.

Наприклад запис на мові С++ умови $x \in [0;100]$ має вигляд:

x >= 0 && x <=100

Увага. Для поліпшення читабельності програми можна ставити дужки і в тих місцях, де вони не обов'язкові, наприклад (x >= 0) && (x <=100).

Операція привласнення

Операція привласнення позначається знаком =.

1. Вираз виду i = i + 2 може бути записаний у вигляді i+=2

   i = i*2 еквівалентно i*=2

   i=i/10 еквівалентно i/=10

   i=i%10 еквівалентно i%=10

2. Багатократне привласнення a = b = c = x*y відбувається справа наліво. Спочатку обчислюється **x*y**, потім його значення привласнюється змінній ***c***, потім ***b*** і лише потім змінній ***a***.

Умовна операція

Загальний вигляд умовної операції:

e1? e2:e3 , де e1, e2, e3 – вирази

Якщо **e1** набуває значення істина (тобто ≠ 0), то значенням цієї конструкції буде **e2**, інакше **e3**.

Наприклад, знаходження більшого з 2-х чисел **a** і **b** може бути записано у вигляді: max = (a > b)? a: b;

Складений оператор (блок) – це декілька операторів, які поміщені у фігурні дужки **{ }**. Блок еквівалентний одному операторові, символ **«;»** після блоку не ставиться.

Порожній оператор. Цей оператор використовується там, де по синтаксису мови потрібен оператор, а по сенсу ніякі дії не виконуються.

Умовний оператор IF

Оператор *if* забезпечує передачу управління на одну з двох гілок обчислень. Розглянемо два варіанти оператора *if*:

а) **if (умова) оператор1;**

Якщо умова набуває значення істина (тобто ≠ 0), то виконується *оператор1*, інакше – наступний оператор програми.

б) **if (умова) оператор1;**
else оператор2;

Якщо умова набуває значення істина (тобто ≠ 0), то виконується *оператор1*, інакше виконується *оператор2*.

Приклад 1. Обчислення значення функції **y** при різних значеннях **x.**

$$y = \begin{cases} \cos x, & x \le 0 \\ \arcsin x, & 0 < x \le \dfrac{\pi}{2} \\ \log_4 x, & \dfrac{\pi}{2} < x \le 64 \\ \dfrac{1}{x^2}, & x > 64 \end{cases}$$

Програма в консольному режимі має вигляд:

```
#include<math.h>
#include<conio.h>
#include<iostream.h>
//---------------------------------------------------------------------------
void main() {
  double x, y;
  clrscr();
```

```
  cout << "ввести значення x";
  cin>>x;
  if (x <= 0) y = cos(x);
  if (x > 0 && x <= M_PI/2)   y = asin(x);
  if (x > M_PI/2 && x <= 64)  y=log(x)/log(4);
  if (x > 64)   y=1./pow(x, 2);
  cout << "y=" << y;
  getch();
}
```

При роботі наведеної вище програми завжди виконуються один за одним всі чотири умовних оператора, при цьому справжнім виявляється тільки один умовний вираз, відповідно, присвоювання значення змінної ***y*** виконується один раз.

Тестові приклади для цієї програми повинні включати принаймні по одному значенню аргументу з кожного інтервалу, а також ще й всі крапки перегину.

Приклад 2. Створення у графічному середовищі Builder проекту для обчислення значення функції **y** при різних значеннях **x**

$$y = \begin{cases} a + b - x, & x \geq 1{,}5 \\ \sin^2 x + \cos(a + b), & x < 1{,}5 \end{cases}$$

$a = 1{,}3 - x^2$; $\mathrm{b} = 0{,}85$

Порядок дій:

– створити новий проект, зберегти його;

– розмістити об'єкти (сторінка Standard палітри компонентів) та задати їм необхідні властивості:

об'єкт **Label1**, властивість *Caption → ЗНАЧЕННЯ АРГУМЕНТУ;*

об'єкт **Label2** для виведення значень **x;**

об'єкт **Label3**, властивість *Caption → ЗНАЧЕННЯ ФУНКЦІЇ;*

об'єкт **Label4**, для виведення значень **y;**

об'єкт **Button1** для запуску проекту, властивість *Caption → ОБЧИСЛИТИ;*

об'єкт **Button2** для очищення полів, властивість *Caption → ОЧИСТИТИ.*

– обробник події *Click* кнопки *ОБЧИСЛИТИ* має вигляд*:*

```
#include <math.h>
```

```
#include <stdlib.h>
//---------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  float b=0.85;
  float x, y, a;
  x=StrToFloat(InputBox("vvedite", "x",""));
  a=1.3 - x*x;
  if(x>=1.5) y=a+b - x;
  else
  y=sin(x)*sin(x)+cos(a+b);
  Label2 ->Caption=x;
  Label4 ->Caption=y;
}
```

– обробник події *Click* кнопки *ОЧИСТИТИ* має вигляд:

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  Label2 ->Caption="";
  Label4 ->Caption="";
}
```

Результат виконання коду надано на рисунку 4.2.

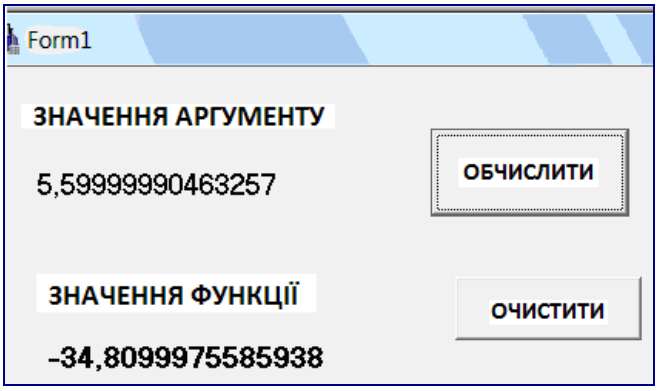


Рисунок 4.2 – Результат виконання коду

Оператор вибору switch

Оператор вибору *switch* (оператор множинних розгалужень**)** забезпечує передачу управління на одну з довільних кількостей гілок.

Синтаксис оператора *switch*:

switch (вираз){
case const1: послідовність операторів; **break;**
case const2: послідовність операторів; **break;**
……
case constN: послідовність операторів; **break;**
[**default**: послідовність операторів; **break;**]
}

Вираз, що стоїть в дужках після ключового слова, а також вирази *const1, const2, …* повинні бути цілочисельного типу.

Правила виконання оператора:

– обчислюється вираз в заголовку і порівнюється послідовно з *const1, const2, …, constN.* Як тільки буде знайдено відповідність, виконаються оператори, що йдуть за двокрапкою;
– якщо відповідність ніде не встановлена, то виконаються оператори, що йдуть за ключовим словом default.

break – вихід з оператора *switch* і перехід до наступного оператора програми.

Розглянемо приклад програми, що забезпечує виведення назв днів тижня по їх номеру.

Консольний режим:

```
#include <conio.h>
#include <iostream.h>
//---------------------------------------------------------------------------
void main() {
  int dn;                              //dn – номер дня тижня
  cout<<"input dn";                    //запрошення до вводу
  cin>> dn;                        //у змінну dn  вводиться номер дня тижня
  switch (dn){
    case  1 : cout<<"Monday"; break;
    case  2 : cout<<"Tuesday"; break;
    case  3 : cout<<"Wednesday"; break;
    case  4  : cout<<"Thursday"; break;
    case   5 : cout<<"Friday"; break;
```

```
    case   6 : cout<<"Satyrday"; break;
    case   7 : cout<<"Sunday "; break;
    default: cout<<" input integer number   1-7 " ; break;
  }
  getch();
}
```

Увага. Оператор *switch* краще оператора *if* в тих випадках, якщо в програмі потрібно розгалузити обчислення на кількість напрямків більше двох і вираз, за значенням якого виробляється перехід на ту чи іншу гілку, є цілочисельним. Рекомендуємо завжди використовувати в операторі *switch* гілку *default.*

Зона дії змінних

Змінна має бути оголошена до її використання. Існують змінні локальні (внутрішні) і глобальні.

Локальна змінна існує тільки в тому блоці, в якому оголошена. При виході з блоку ця змінна і її значення втрачаються. Зона дії локальної змінної – блок.

Наприклад:

```
for ( int i=0; i<10; i++)
  {тіло циклу}
i=0;
```

Перша змінна *i* відома тільки в циклі *for*, а у виразі *i=0***;** це буде вже інша змінна, тобто їй компілятор присвоїть зовсім іншу адресу.

Глобальна змінна – це змінна, оголошена поза якою-небудь функцією і може бути використана у будь-якому місці програми.

Зона дії глобальної змінної – уся програма.

Наприклад:

```
int a;
        int main() {
        ……..
        }
```

Увага. Віддавайте перевагу локальним змінним перед глобальними. Змінна повинна мати мінімальну з можливих областей дії.

4.3 Циклічний обчислювальний процес

Циклічними називаються обчислювальні процеси, в яких неодноразово виконуються одні й ті ж дії, але з різними даними. Обчислювальний процес, що містить один або кілька циклів, називається циклічним. Цикл – це послідовність операторів, повторювана багаторазово. В мові С++ існує три види взаємозамінних операторів циклу*: for, while, do … while*.

Оператор циклу FOR

Оператор *for* зручний при організації циклів, коли кількість повторень відома. Загальний вигляд оператора :

for (вираз1; вираз2; вираз3) тіло циклу;

вираз1 – задає початкове значення параметра циклу;

вираз2 – задає умову продовження циклу ;

вираз3 – задає зміну параметра циклу;

тіло циклу – може бути або простій, або складений оператор.

Алгоритм роботи циклу:

1. Привласнюється значення параметру циклу.
2. Перевіряється умова продовження циклу. Якщо умова набуває значення *істина* (≠0), то виконується тіло циклу і перехід на пункт 1, інакше виконується оператор, що йде за оператором *for***.**

Таким чином, перед кожним повторенням циклу відбувається зміна параметра циклу і перевірка умови завершення циклу.

Приклад 1. for (i= 0; i < 10; i ++) cout << i << "\n";

В результаті роботи циклу у стовпець виведуться цифри від 0 до 9.

Приклад 2. for (i = 9; i >= 0; i --) cout << i << "\n";

В даному випадку в стовпець виведуться цифри від 9 до 0.

Приклади нескінченних циклів (можуть бути отримані випадково):

for (i = 1; 1; i ++) оператор;

for (i=10; i>6; i++) оператор;

У даних випадках умови продовження циклу завжди мають значення *істина* (≠0).

Приклад 3. Побудова таблиці значень функції

$$z = \begin{cases} \alpha x^2 + \lambda, & x > \lambda \\ \alpha y + x, & x \le \lambda \end{cases}$$

$$x = 3\lambda + \cos\lambda$$

$$y = 2\sin\lambda, \quad \alpha = 3{,}0; \quad \lambda \in [-5;5]; \quad \Delta\lambda = 0{,}5$$

Порядок дій (графічний режим):

– створити і зберегти проект;

– для виведення результатів у вигляді таблиці необхідно встановити на формі компонент **StringGrid** (сторінка **Additional**). Визначити кількість рядків в таблиці за формулою $\left[\frac{\alpha_к - \alpha_п}{\Delta\alpha}\right] + 1 = 21;$

– задати властивості компонента *StringGrid:*

ColCount → 2 (кількість стовпців);

RowCount → 21 (кількість рядків);

– встановити у форму два компоненти *Label* і компонент *Button*, дати їм заголовки відповідно.

Обробник події *Click* кнопки *ОБЧИСЛИТИ* має вигляд:

```
#include<math.h>
void __fast call TForm1::Button1Click(TObject *Sender)
{
    double x, y, ln=-5, lk=5, dl=0.5, z, a=3.0; //оголошення, ініціалізація змінних
    int i = 0;                                  //завдання початкового номера рядка
    for (double l = ln; l <= lk; l += dl)  {
        x=3*l+cos(l);
        y=2*sin(l);
        if (x > l) z = a*x*x+l;
        else
        z = a*y+x;
        StringGrid1 –>Cells[0][i]=l;            //Cells[Col][Row]
        StringGrid1 –>Cells[1][i]=z;            //виведення результатів в таблицю
        i++;                                    //зміна номера рядка в таблиці
        }
}
```

В даному прикладі областю видимості локальної змінної *l* є тільки цикл. Усі результати обчислень можуть бути переглянуті за допомогою вертикальної лінійки компонента *StringGrid* (рис.4.3).

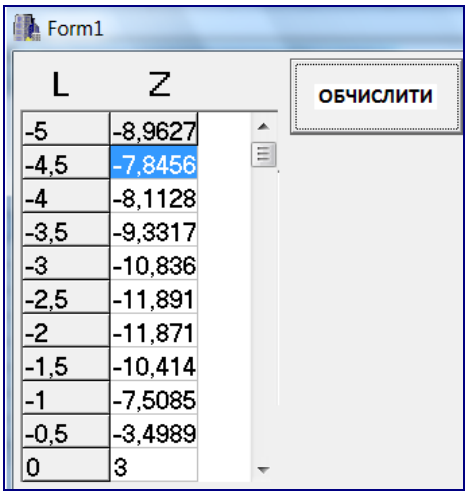


Рисунок 4.3 – Результат виконання коду

Увага. Укладайте в блок (фігурні дужки) тіло циклів, якщо в них потрібно виконати більше одного оператора.

Обчислення суми ряду

Приклад. Обчислення суми ряду

$$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots+\frac{1}{49}-\frac{1}{50}$$

Програма в консольному режимі має вигляд:

```
#include<iostream.h>
#include<conio.h>
//---------------------------------------------------------------------------
void main() {
    double s=0;                          //ініціалізація суми
    for (int i=1; i<=50; i++)
    s+=pow(–1, i+1) / i;                 //накопичення суми
    cout<<"s="<<s;                       // виведення суми ряду на екран
    getch();
}
```

Результат S=0,683247.

Оператор циклу WHILE

Оператор циклу *while* зручний в тих випадках, коли або число ітерацій заздалегідь невідомо, або очевидних параметрів циклу немає, або модифікацію параметрів зручніше записувати десь в довільному місці.

Загальний вигляд оператора *while*:

while (умова продовження циклу) тіло циклу;

Тіло циклу – простий оператор або складений.

Цикл виконується до тих пір, поки умова набуває значення *істина* (≠0). Якщо умова приймає значення *неправда*, то управління передається наступному операторові програми.

Так само як і в циклі *for*, спочатку перевіряється умова, а потім виконується тіло циклу. Це цикли з передумовою.

Приклад. Для заданого рядка (прізвище користувача, задане латинськими буквами) відображаються ASC коди кожного символу.

У C++ рядок − це масив символів. Ознака кінця рядка – '\0' (нульовий байт), який автоматично додається компілятором.

Нехай змінна *name* – це прізвище користувача.

Алгоритм рішення задачі у консольному режимі має вигляд:

```
#include<iostream.h>
#include<conio.h>
void main() {
    char name [10];                          //довжина масиву з урахуванням '\0'
    cout << "input fam"<<endl;
    cin >> name;                             //введення рядка
    cout << " simv, kod"<<"";
    int i=0;                                 //початкове значення параметра циклу
    while (name[i]!='') {
        cout <<name[i]<<"   :"<< int(name[i])<<"\n";
        i++;                                 //зміна параметра циклу
    }
    getch();
}
```

Цикл працюватиме, доки не зустрінеться ознака кінця рядка '\0'.

Результат роботи програми надано на рисунку 4.4.

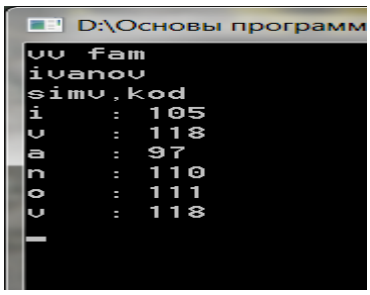


Рисунок 4.4 – Результат виконання коду

Оператор циклу DO … WHILE

Оператор циклу *do…while* використовують в тих випадках, коли цикл необхідно виконати обов'язково хоча б один раз – наприклад, при введенні даних.

Загальний вигляд оператора:

do

тіло циклу

while (умова продовження циклу **);**

Це цикл з постумовою, тобто спочатку виконується тіло циклу, а потім оцінюється умова продовження циклу. Якщо в результаті оцінки виходить значення *істина*, то цикл повторюється. Якщо отримано значення *неправда*, то цикл завершується.

Таким чином, цикл виконається хоча б одного разу.

Приклад. Обчислення значення функції e^x за допомогою нескінченного ряду Тейлора з точністю до **ε** за формулою

$$e^x = 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots + = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

Обчислення проводиться до тих пір, поки не виконається умова $\frac{x^n}{n!} \le \varepsilon$, де **ε** –як завгодно мала величина.

Цей ряд сходиться при $|x| < \infty$. Для досягнення заданої точності будемо підсумовувати члени ряду. При деякому значенні n умова $\frac{x^n}{n!} \le \varepsilon$ виконається і обчислення припиняються.

Алгоритм рішення в консольному режимі має вигляд:

```
#include<conio.h>
#include<iostream.h>
```

```
#include<math.h>
//---------------------------------------------------------------------------
int main() {
  float x, y, s=1, eps, p=1;          //перший член ряду s=1
  int n=1;                            //початкове значення параметра циклу
  cout <<"\n input  x i eps\n" ;
  cin>>x>>eps;                        //введення з клавіатури
  do   {
       p=p*n;
       y=pow(x, n)/p;                 //загальний член ряду
       s+=y;                          //накопичення суми ряду
       n+=1;                          //зміна параметра циклу
   }
  while(y>=eps);                      //перевірка умови продовження циклу
  cout<<"summa="<<s<<endl;
  cout<<"element rayda="<<y;          //наступний член ряду
  getch();
}
```

Результат роботи програми надано на рисунку 4.5.

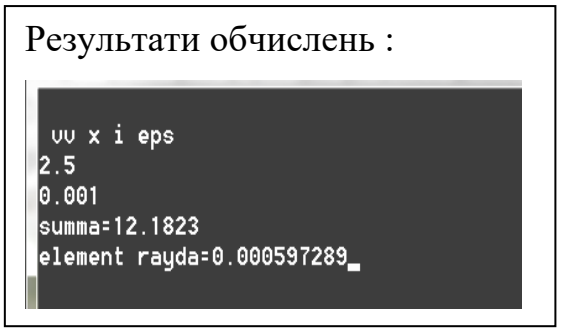


Рисунок 4.5 – Результат виконання коду

4.4 Одновимірні масиви

Масив – впорядкований набір фіксованої кількості однотипних елементів, що зберігаються в послідовно розташованих комірках оперативної пам'яті, мають порядковий номер і спільне ім'я, що надає користувач. Масив може бути одновимірним (вектором), та багатовимірним, тобто таким, де індексом є не

одне число, а кортеж (сукупність) з декількох чисел, кількість яких збігається з розмірністю масиву.

Генерація випадкових чисел

Функція **rand()** генерує ціле число в діапазоні між **0** і символьною константою **RAND_MAX,** визначеною в заголовному файлі **<stdlib.h>**. Для того, щоб вибрати цілі числа у конкретному діапазоні, використовується операція обчислення залишку **%.**

Наприклад:

rand()%n	генеруються цілі випадкові числа з [0; n–1];
a+rand()%(b – a+1)	генеруються цілі випадкові числа з [a; b];
(double) rand()/RAND_MAX	генеруються дійсні випадкові числа з [0; 1].

Спільно з функцією *rand* використовується бібліотечна функція **srand**, яка ініціалізує генератор випадкових чисел від таймера:

srand (time (NULL)).

Функція ***time*** повертає поточний час в секундах. В результаті, при кожному запуску програми генеруються різні числа.

Приклади введення одновимірних масивів.

Приклад 1. Введення масиву A(10) з клавіатури і вивід його на екран:

```
for (i=0; i<10; i++) cin>>a[i];          //нумерація елементів з нуля
for (i=0;I <10; i++) cout << a[i];
```

Приклад 2. Генерація одновимірного масиву розмірністю 7 цілих чисел в діапазоні [-10;10]. Графічний режим:

```
srand(time(NULL));
for (int i=0; i<7; i++){
   x[i]= –10 +rand()%21;          //заповнення масиву випадковими числами
   StringGrid1 –>Cells[i][0] =x[i];   //виведення масиву в рядок
  }
```

Приклад 3. Заповнення одновимірного масиву розмірністю 10 цілими випадковими числами з діапазону [-20;20]. Пошук середнього арифметичного від'ємних елементів масиву.

Для знаходження середнього арифметичного елементів потрібно знайти їх загальну суму, після чого розділити її на кількість елементів.

Реалізуємо алгоритм в графічному середовищі.

Порядок дій. Розмістити об'єкти згідно з рисунком 4.6, задати їм необхідні властивості:

– *об'єкт StringGrid1 (сторінка Addition), властивості:*

ColCount → 10 (кількість стовпців); **RowCount** → 1 (кількість рядків);

– об'єкт **Button1** генерація масиву, властивість *Caption→ЗГЕНЕРУВАТИ МАСИВ;*

– об'єкт **Button2** для запуску проекту, властивість *Caption → ОБЧИСЛИТИ;*

– об'єкт **Label2,** властивість *Caption → СЕРЕДНЄ АРИФМЕТИЧНЕ*;

– об'єкт **Label3** для виведення результату.

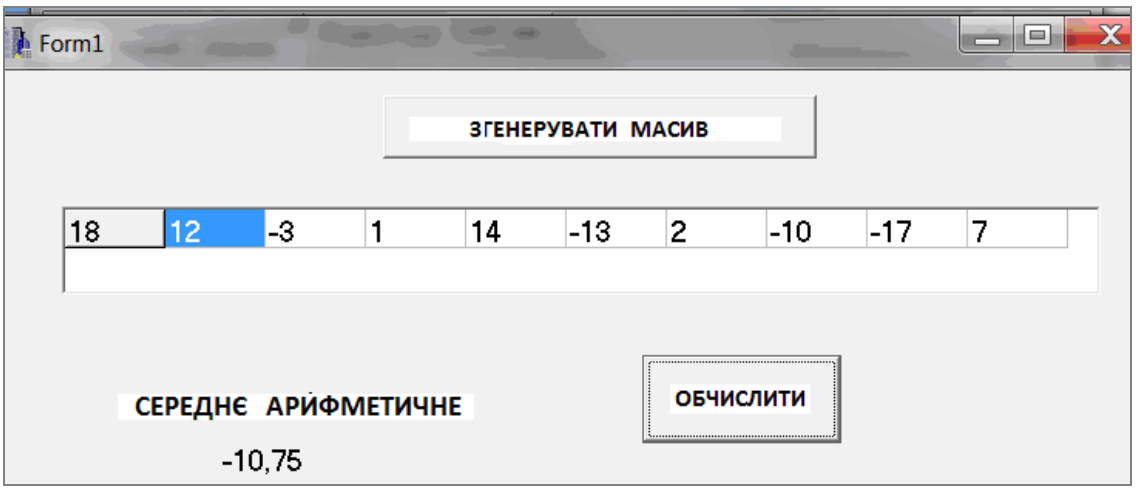


Рисунок 4.6 – Розміщення об'єктів на формі

Обробник події *Click* кнопки *ЗГЕНЕРУВАТИ МАСИВ* має вигляд*:*

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    srand(time(NULL));
    for (int i=0;i<=9;i++){
      x[i]= –20 +rand()%41;          //заповнення масиву випадковими числами
      StringGrid1 –>Cells[i][0] =x[i];        //виведення масиву
    }
}
```

Обробник події *Click* кнопки *ОБЧИСЛИТИ* має вигляд*:*

```
TForm1 *Form1;
int x[10];                    //масив оголошується як глобальний параметр
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  double sa;                  //оголошення змінної середнє арифметичне
```

```
int i, sum=0, k=0;              //ініціалізація змінних сума і кількість
for (int i=0;i<=9;i++)
    if(x[i]<0){                 //пошук від'ємних елементів масиву
    sum+=x[i];                  //накопичення суми
    k++;                        //накопичення кількості
    }
sa=(double) sum/k;              //знаходження середнього арифметичного
Label3 –>Caption=sa;
}
```

Ця програма працює вірно для масивів, що містять від'ємні елементи. Якщо від'ємні елементи відсутні, то вона завершується аварійно в рядку sa=(double) sum/k. Тому потрібна спеціальна обробка цієї ситуації.

Для виконання необхідної перевірки змінній *neg* призначимо від'ємне значення (наприклад, neg= –1), це буде говорити про те, що від'ємних елементів в масиві немає. Як тільки в масиві зустрінеться від'ємний елемент, значення змінної *neg* буде змінено (наприклад, neg= 1).

Нижче представлений виправлений варіант обробника події *Click* кнопці *обчислити*.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  double sa;
  int i,sum=0,k=0;
  int neg= –1;
  for (int i=0;i<=9;i++)
    if(x[i]<0) {
    sum+=x[i];
    k++;
    neg=1;
  }
  if (neg== –1)  Label3->Caption="no elements";
  if (neg==1){
  sa=(double)sum/k;
  Label3->Caption=sa;
  }
}
```

Динамічний розподіл пам'яті

Якщо розмірність масиву задається під час виконання програми, наприклад, введена з клавіатури, то такі масиви називаються *динамічними*. Пам'ять під них виділяється за допомогою оператора **new.** Наприклад, нехай

arr – ім'я масиву;

***arr** – адреса масиву;

n – розмірність масиву (кількість елементів),

тоді у консольному режимі:

```
cin>>n;                    //ввод розмірності масиву
int *arr=new int[n];
```

В даному випадку, за адресою масиву з ім'ям ***arr***, операцією **new** буде виділена безперервна область пам'яті. Пам'яті виділиться стільки, скільки необхідно для зберігання *n* величин типу *int.*

Увага. Обнуління пам'яті при її виділенні не відбувається. Ініціалізувати динамічний масив не можна.

Приклад. Створення проекту для генерації масиву цілих чисел будь-якої розмірності. Обчислення суми елементів з номерами кратними трьом і суми інших елементів масиву.

Якщо точна кількість елементів в початковому масиві не задано, але відомо, що вона не може перевищувати деяке конкретне значення, то у цьому випадку кількість рядків в таблиці виводу виділяється по «максимуму».

Реалізуємо алгоритм в графічному середовищі.

Порядок дій :

1. Встановити об'єкти згідно з рисунком 4.7.

– *Label1, Label2* – для виведення результатів;

– *Button1* – для запуску проекту, властивість *Caption→ЗГЕНЕРУВАТИ МАСИВ ТА ОБЧИСЛИТИ;*

– *StringGrid1* – для виведення елементів масиву:

ColCount→1(кількість стовпців);

RowCount→(кількість рядків);

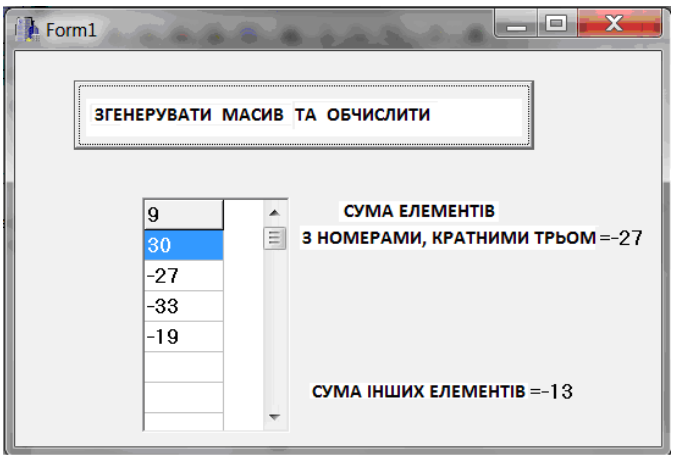


Рисунок 4.7 – Розміщення об'єктів на формі

2. Обробник події *Click* кнопки *ЗГЕНЕРУВАТИ МАСИВ ТА ОБЧИСЛИТИ* має вигляд:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    int s1=0, s2=0;
    int size;                                   //оголошення розмірності масиву
    size=StrToFloat(InputBox("vv", "size",""));   //введення розмірності масиву
    int*arr=new int[size];                      //динамічне виділення пам'яті
    srand (time(NULL));
    for ( int i=0;i<size;i++){
        arr[i]=rand()%100-50;           //заповнення масиву випадковими числами
        StringGrid1 –>Cells[0][i]=arr[i];              //виведення масиву
    }
    for (int i=0; i<size ;i++)
        if((i+1) %3==0 )           //пошук елементів з номерами кратними 3
        s1=s1+arr[i];                   //сума елементів з номерами кратними 3
        else
        s2=s2+arr[i];                   //сума інших елементів
    Label1 –>Caption="сума елем. з ном. кратними   3 ="+FloatToStr(s1);
    Label2 –>Caption="сума інших елементів="+FloatToStr(s2);
}
```

Увага. Звернення до елемента динамічного масиву виконується так само, як і до елемента звичайного, наприклад a[3].

4.5 Двовимірні масиви

В мові С++ є можливість використовувати двовимірні та багатовимірні масиви. Двовимірний масив – це список одновимірних масивів.

Для доступу до елементів двовимірного масиву потрібно вказати два індекси. Якщо розглядати масив як таблицю, тоді перший індекс визначає рядок. Другий індекс визначає стовпець таблиці.

Приклад. Заповнення двовимірного масиву A(4,5) випадковими цілими числами з діапазону [–10;10]. Обчислення суми елементів в кожному рядку.

Створення проекту в графічному середовищі.

Порядок дій:

1. Розмістити об'єкти згідно з рисунком 4.8, задати їм такі властивості:

- об'єкт **Label1,** властивість *Caption → ПОЧАТКОВИЙ МАСИВ;*
- об'єкт **Label2,** властивість *Caption → СУМА ЕЛЕМЕНТІВ ПО СТРОКАХ;*
- об'єкт **StringGrid1** для виведення елементів масиву у вигляді таблиці: властивості **ColCount** → 5 (кількість стовпців);
  RowCount → 4 (кількість рядків);
- об'єкт **StringGrid2** для виведення результатів:
  властивості **ColCount** → 1 ;
  RowCount → 4 ;
- об'єкт **Button1** для запуску проекту, властивість *Caption→ ОБЧИСЛИТИ.*

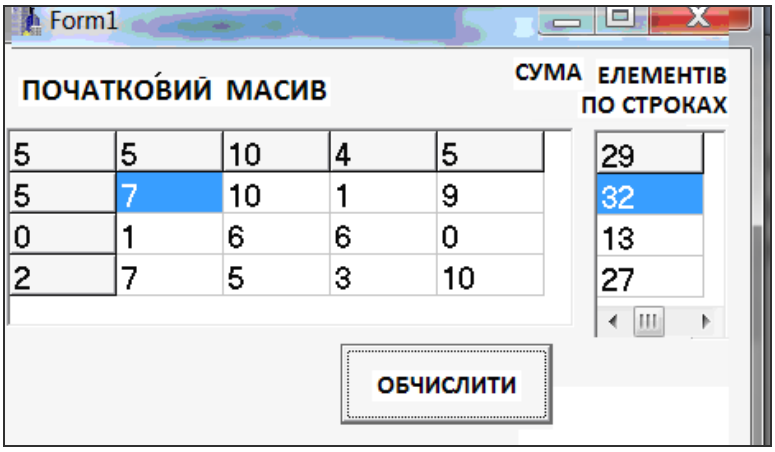


Рисунок 4.8 – Розміщення об'єктів на формі

2. Обробник події Click кнопки *ОБЧИСЛИТИ* має вигляд:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  const int nrow=4, ncol=5;       //розмірність масиву, іменовані константи
```

```
  int a[nrow][ncol];                    //оголошення масиву
  int s;
  srand(time(NULL));              //ініціалізація генератора випадкових чисел
  for (int i=0; i<nrow; i++)
    for (int j=0; j<ncol; j++)  {
        a[i][j]=rand()%21–10;          //заповнення масиву випадковими числами
        StringGrid1 ->Cells[j][i]=a[i][j];          //заповнення таблиці числами
  }
  for (int i=0; i<nrow; i++) {
    s=0;
    for (int j=0;j<ncol; j++)
        s+=a[i][j];                          //накопичення суми в рядку
    StringGrid2 ->Cells[0][i]=s;       //виведення результатів в таблицю
  }
}
```

У даному коді розмірність масиву задається іменованими константами, що дозволяє їх легко змінювати.

4.6 Функції користувача в C++

Функція – це іменована група операторів, що виконують закінчену дію. До неї можна звернутися по імені, передати їй значення і отримати з неї результат.

З використанням функції пов'язано три поняття:

- оголошення функції;
- визначення функції – опис дій, які виконує функція;
- звернення до функції.

Оголошення функції

Оголошення функції задається її заголовком.

У заголовку вказується ім'я функції, тип результату, який вона повертає, скільки аргументів і якого типу їй потрібно передати. Формат найпростішого заголовка (прототипу) функції:

[тип] ім'я функції ([список формальних параметрів]);

У квадратних дужках записано те, що може бути опущено.

До першого виклику функції необхідно повідомити тип результату, який очікується, а так само кількість і типи аргументів (це необхідно для виділення місця в пам'яті). У програмі може міститися довільна кількість оголошень однієї і тієї ж функції.

Наприклад, оголошення функції пошуку максимального з двох цілих чисел:

max (int a, int b); або

max (int, int); /* імена формальних параметрів не грають ніякій ролі і ігноруються компілятором*/

Увага. Все, що передається в функцію і назад, повинно відображатись в її заголовку.

Визначення функції

Визначення функції крім заголовка включає її тіло, тобто оператори, які виконуються при її виклику:

[тип] ім'я функції ([список формальних параметрів]){

тіло функції

}

Тип – це тип значення, яке повертається. Якщо тип не заданий, то за умовчанням мається на увазі тип *int*. Якщо функція не повертає значення, то тип *void*.

Формальні параметри – це змінні, використовувані усередині тіла функції. Вони набувають значень при виклику функції шляхом копіювання в них значень відповідних фактичних параметрів. Для кожного формального параметра має бути вказаний тип.

Для виходу з функції і повернення результату обчислень у точку виклику служить оператор

return [вираз];

У тілі функції може бути декілька операторів *return* або жодного.

Приклад. Функція користувача для знаходження більшого з двох цілих чисел. Можливі варіанти:

а)
```
max (int a, int b){
if (a>b) return a;
else return b;
}
```

б)
```
max (int a, int b){
 return (a>b)? a: b;
}
```

Звернення до функції

Для виклику функції потрібно вказати її ім'я, а також передати їй набір аргументів відповідно до її заголовка:

ім'я функції ([список фактичних параметрів])

Увага. У визначенні, в оголошенні і при виклику функції типи і порядок проходження параметрів повинні співпадати.

Приклад. Створення функції, що повертає суму елементів масиву. Скористатися нею при обчислення сум елементів масивів X і Y, що містять 10 і 3 елементи відповідно.

Для нашої функції знаходження суми елементів масиву необхідно отримати ззовні масив елементів і кількість цих елементів. Нехай масив і результат мають тип *float*, а кількість елементів тип *int*. Отже, заголовок функції може виглядати так:

float sum(float a[], int n);

Створимо проект в графічному середовищі.

Порядок дій:

– Розмістити об'єкти згідно з рисунком 4.9, задати їм необхідні властивості.

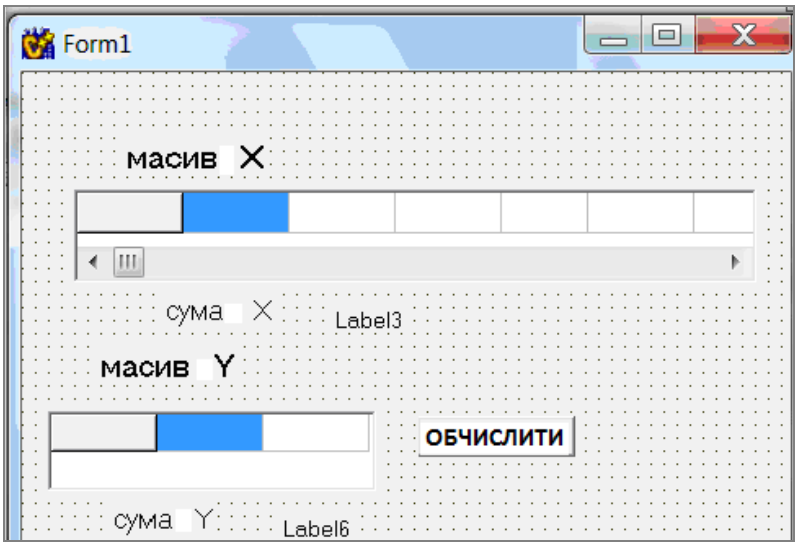


Рисунок 4.9 – Розміщення об'єктів на формі

– Обробник події *Click* кнопки *ОБЧИСЛИТИ* має вигляд:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float sum(float a[], int n);                    //оголошення функції
```

```
    float x[10]={7.8, 6, 9, 0, 5.5, 6.6, 7, 8, 9, 10};
    float y[3]={1.1, 2.2, 3.3};
    for (int i=0; i<=9; i++)
    StringGrid1 –>Cells[i][0]=x[i];                  //виведення масиву x
    Label3 –>Caption=sum(x, 10);                     //звернення до функції
    for (int i=0; i<=2;i++)
      StringGrid2 ->Cells[i][0]=y[i];                //виведення масиву y
    Label6 ->Caption=sum(y, 3); );                   //звернення до функції
    }
    float sum(float a[], int n){                     //визначення функції
    float s=0;
    for (int i=0; i<n;i++)
      s+=a[i];                                       //накопичення суми елементів масиву
    return s;                                        //повернення результату
}
```

Рекурсивні функції

Функція називається рекурсивною, якщо оператор в тілі функції містить виклик цієї ж функції. У мові С++ функції можуть викликати самі себе.

Приклад класичної рекурсивної функції – це обчислення факторіалу числа. Обчислимо n! при $n \in [0; 10]$. Пам'ятаємо, що n!=1*2*3*.*n.

Визначення функції:

```
long int factorial(int n){
if (n==0 | n==1) return 1;
return factorial (n – 1)*n;                  //функція викликає сама себе
}
```

Проект на С++ в консольному режимі має вигляд:

```
#include<iostream.h>
#include<conio.h>
//……..
long int factorial (int n);                          //це оголошення функції
void main() {                                        //це головна функція
    for (int i=0; i<=10; i++)
    cout<<i<<"!="<<factorial(i)<<"";                 //звернення до функції
    getch();
```

```
}
long int factorial(int n){                    //це визначення функції
if (n==0 | n==1) return 1;
return factorial(n – 1)*n;
}
```

Зазвичай заголовки всіх функцій (оголошення) розміщуються на самому початку файлу

Результат роботи програми надано на рисунку 4.10.

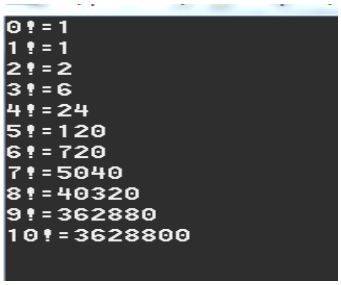


Рисунок 4.10 – Результат роботи програми

4.7 Операції над двійковим кодом

У обчислювальній техніці уся інформація (числа, текст, графічні дані, звук) кодується двійковим кодом у вигляді {0; 1}. Коди чисел від 1 до 16 в десятковії та двійковій системах представлені нижче.

Decimal	Binary
1	0000 0001
2	0000 0010
3	0000 0011
4	0000 0100
5	0000 0101
6	0000 0110
7	0000 0111
8	0000 1000

Decimal	Binary
9	0000 1001
10	0000 1010
11	0000 1011
12	0000 1100
13	0000 1101
14	0000 1110
15	0000 1111
16	0001 1111

Порозрядні логічні операції, зсуви

Порозрядні логічні операції виконуються над відповідними бітами цілих чисел. Кожен біт має значення 0 або 1, що наочно демонструє **таблиця істинності**:

. **Значення бітів**		**Результат операції**			
e1	**e2**	**e1 & e2**	**e1 | e2**	**e1 ^e2**	**~e1**
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

1 – істина; 0 – неправда.

Порозрядними логічними операціями є:

& порозрядне логічне І;

| порозрядне логічне АБО;

^ порозрядне, що виключає АБО;

~ порозрядне заперечення;

<< зсув вліво;

>> зсув вправо.

Порозрядні логічні операції дозволяють забезпечити доступ до кожного біта інформації.

Приклад 1. Використання порозрядного логічного І.

Результат набуває значення 1, якщо обидва біти мають значення 1.

У 2-ій системі числення 1100 0001

& 0100 0010

0100 0000

Ми бачимо, якщо необхідно встановити значення розряду, що дорівнює нулю, то користуються операцією &.

Приклад 2. Використання порозрядного логічного АБО.

Результат набуває значення 1, якщо хоч би один з бітів має значення 1.

У 2-ій системі числення 1100 0001

| 0100 0010

1100 0011

Ми бачимо, якщо необхідно встановити значення розряду, що дорівнює одиниці, то користуються операцією |.

Приклад 3. Використання порозрядного, що виключає АБО (^)

Результат набуває значення 1, якщо значення тільки одного з бітів дорівнює 1 (один і тільки один).

У 2-ій системі числення 1100 0001

^ 0100 0010

1000 0011

Приклад 4. Логічне заперечення (~).

У 2-ій системі числення : ~0100 0001 → 1011 1110

Пріоритети виконання операцій мови С++ представлені в таблиці 4.1.

Таблиця 4.1 – Пріоритети операцій мови С++

Операція	**Призначення**
++ – –	збільшення (зменшення) на 1
~	порозрядне заперечення
!	логічне НІ
+ -	унарний +, унарний –
( )	приведення типу
* / %	множення, ділення, обчислення
+ –	складання, віднімання
>> <<	зсуви
< <= > >= == !=	операції відношення
&	порозрядне логічне І
^	порозрядне, що виключає АБО
|	порозрядне логічне АБО
&&	логічне І
||	логічне АБО
?:	умовна операція
= += *=	привласнення

4.8 Завдання для самоконтролю та приклади їх виконання

Розробити проекти у середовищі C++ Builder для рішення задач.

1. Обчислити значення функції

$$y = \frac{x^4 - bx^3 - a}{(x+a)(x-b)},\ де\ b = 2^x;\ x = \lg 0{,}05 + 2;\ a = b + 2.$$

2. Обчислити значення функцій за двома формулами

$$z1 = 2\sin^2(3\pi - 2\alpha)\cdot\cos^2(5\pi + 2\alpha);$$

$$z2 = \frac{1}{4} - \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right).$$

Значення α ввести з клавіатури. Результат обчислення за обома формулами має бути однаковим.

3. Обчислити температуру за Цельсієм, якщо вона задана в градусах за Фаренгейтом, згідно з формулою:

C=5/9*(F-32),

де C – температура за Цельсієм, а F – температура за Фаренгейтом.

4. Задано ціле число. Визначити, чи є воно парним.

5. Обчислити значення функції Z при різних значеннях аргументу x.

$$z = \begin{cases} x^2 - \sin\gamma, & x \le 0 \\ \sqrt{x} + \cos\gamma, & x > 0 \end{cases},$$

де γ=0.35α; α=x+3.

6. Знайти мінімум з трьох чисел x, y, z ($x \neq y \neq z$).

7. Обчислити $\frac{1}{1\cdot 2} + \frac{1}{2\cdot 3} + \frac{1}{3\cdot 4} + \ldots + \frac{1}{999\cdot 1000}$.

8. Обчислити функцію

$$y = \begin{cases} t & , x < 0 \\ tx & , 0 \le x < 2 \\ 2t & , x \ge 2 \end{cases},$$

де $x \in [-3;\ 3{,}5];\quad \Delta x = 0{,}3$

Результат вивести у вигляді таблиці.

9. Обчислити значення функції

$$Z=\begin{cases} x^2\sin\alpha+y^2 & y>1 \\ \sqrt{x^2+y^2}\,\mathrm{tg}\,\alpha & y\le 1 \end{cases},$$

де $x=a^2e^{1.5\alpha}; y=3.8\ln 5\alpha; a=3,4; \alpha_н=0,30; \alpha_к=0,60; \Delta\alpha=0,05.$

10. Обчислити $\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)\ldots\left(1+\frac{1}{n^2}\right)$, де n – задане число.

11. Обчислити суму ряду $\frac{x}{3}+\frac{x^2}{6}+\frac{x^3}{9}+\ldots+\frac{x^9}{27}$, де x – задане число.

12. Обчислити значення функції з точністю до ε за формулою:

$$\ln(x+1)=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^{n+1}}{n+1}=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\ldots,$$ якщо $-1<x<1.$

13. Знайти суму елементів масиву з непарними номерами. Масив містить N дійсних величин.

14. Знайти максимальний та мінімальний елементи масиву. Масив містить N дійсних величин.

15. Знайти суму модулів елементів масиву, розташованих після першого від'ємного елемента. Масив містить N дійсних величин.

16. Знайти середнє арифметичне елементів, значення яких належать інтервалу (-273; 20) і їх кількість. Масив містить N дійсних величин.

17. Елементи масиву $\boldsymbol{x=(x_1,x_2,\ldots x_n)}$ задані формулою

$\boldsymbol{x_i=i^2}+\sin\boldsymbol{i}$. Знайти $max(\boldsymbol{x_2,x_4,x_6}\ldots)$.

18. Знайти суму елементів масиву, розташованих після максимального елемента. Масив містить N дійсних величин.

19. Знайти суму елементів масиву, розташованих між максимальним і мінімальним елементами.

20. Послідовність елементів $a_1,a_2,\ldots a_n$ задана формулою $a_i=i^2-1$. Визначити суму елементів, кратних 3, і самі елементи.

21. Обчислити $\sum_{i=1}^{25}(b_i - 3{,}6)^2$,

де $b_i = \begin{cases} i^2, & \text{якщо } i - \text{непарне} \\ i^3, & \text{в противному разі} \end{cases}$.

22. Елементи кожного стовпця матриці Y розміром 4×6 поділити на останній елемент стовпця. Отриманий масив вивести.

23. У заданому масиві X(3, 5) замінити всі від'ємні елементи мінімальним елементом масиву. Отриманий масив вивести.

24. Дана цілочисельна прямокутна матриця. Визначити кількість рядків, які не містять жодного нульового елемента.

25. Переписати від'ємні елементи масиву $Y(20)$ в масив $\boldsymbol{Z}$, а інші елементи в масив L.

26. Записати в масив $\boldsymbol{Y}$ підряд 5 перших від'ємних елементів масиву $X(25)$.

27. Записати в масив $\boldsymbol{Y}$ ті елементи масиву $D(20)$, які розташовані до мінімального елемента.

28. Створити функцію для знаходження суми додатних елементів масиву. Скористатися нею при знаходженні сум додатних елементів масивів X та Y, які мають 15 та 10 елементів відповідно.

29. Створити функцію користувача для обчислення площі круга радиуса r. Скористатися функцією для обчислення площі круга для $r \in [1; 15]$, $\Delta r = 2{,}5$.

30. Створити функцію користувача для обчислення довжини дуги кола радіуса r. Скористатися функцією для обчислення довжини дуги кола при $r \in [2; 10]$, $\Delta r = 0{,}5$.

Приклади виконання завдань для самоконтролю

Приклад 1. Заповнити одномірний масив A розмірністю 10 цілими випадковими числами із діапазону [-100; 100]. Із масиву A сформувати:
а) масив B, із додатних елементів;
б) масив C, із парних елементів.
Створити проект в графічному середовищі.

Порядок дій. Створити новий проект, зберегти його. Розмістити об'єкти згідно з рисунком 4.11, додати їм необхідні властивості.

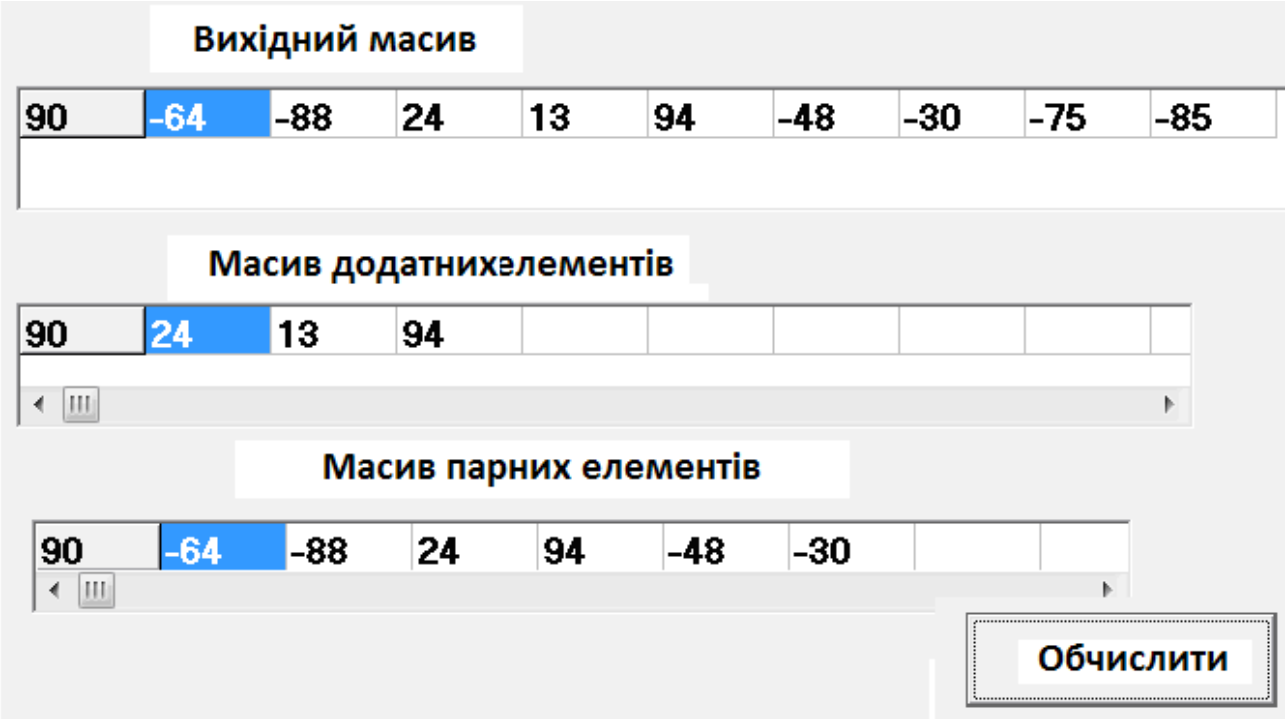


Рисунок 4.11 – Розміщення об'єктів на формі

Оброблювач події Click кнопки *Обчислити* має вигляд:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  int a[10], b[10], c[10];                          //оголошення масивів
  int k=0, l=0;                        //ініціалізація індексів елементів в масивах B і C
  srand(time(NULL));                          //ініціалізація генератора випадкових чисел
  for (int i=0; i<=9; i++) {
    a[i] = rand()%200 – 100;                        //заповнення масиву A
    StringGrid1–>Cells[i][0] = a[i];                //вивід масиву A в таблицю
    if(a[i]>0) {
    b[k]=a[i];                                      //заповнення масиву B
    StringGrid2->Cells[k][0] = b[k];                //вивід масиву B в таблицю
    k++;                                            //формування індексу в масиві B
  }
  if (a[i]%2 == 0) {                                //пошук парних елементів
    c[l]=a[i];                                      //заповнення масиву C
    StringGrid3->Cells[l][0] =c[l];                 //вивід масиву C в таблицю
    l++;                                            //формування індексу в масиві C
    }
  }
```

}

Приклад 2. Для матриці 4×4 обчислити кількість додатних елементів, які знаходяться вище головної діагоналі. Елементи матриці є випадкові числа у діапазоні [-20; 20].

Порядок дій. Створити новий проект, зберегти його. Розмістити об'єкти згідно з рисунком 4.12, додати їм необхідні властивості.

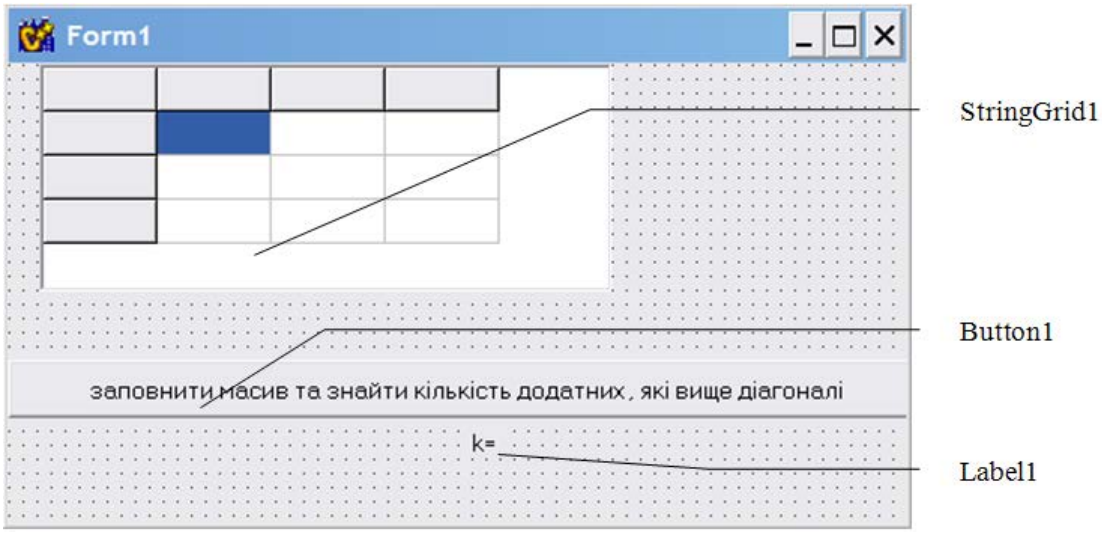


Рисунок 4.12 – Розміщення об'єктів на формі

Оброблювач події *Click* кнопки Button1 має вигляд*:*

```
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
  const int nrow=4, ncol=4;                          //розмір масиву
  int a[nrow][ncol];              //оголошення масиву іменованими константами
  int  k=0;                                          //кількість додатних елементів
  srand(time(NULL));                 //ініціалізація генератора випадкових чисел
  for (int i=0; i<nrow; i++)
  for (int j=0; j<ncol; j++) {
     a[i][j]= rand()%41 – 20;                        //заповнення масиву A
     StringGrid1->Cells[j][i] = a[i][j];          //вивід масиву A в таблицю
     if (i<j & a[i][j]>0) k++;                       //обчислення кількості елементів
     Label1–>Caption = "k="+IntToStr(k);
  }
}
```

Результат роботи коду надано на рисунку 4.13.

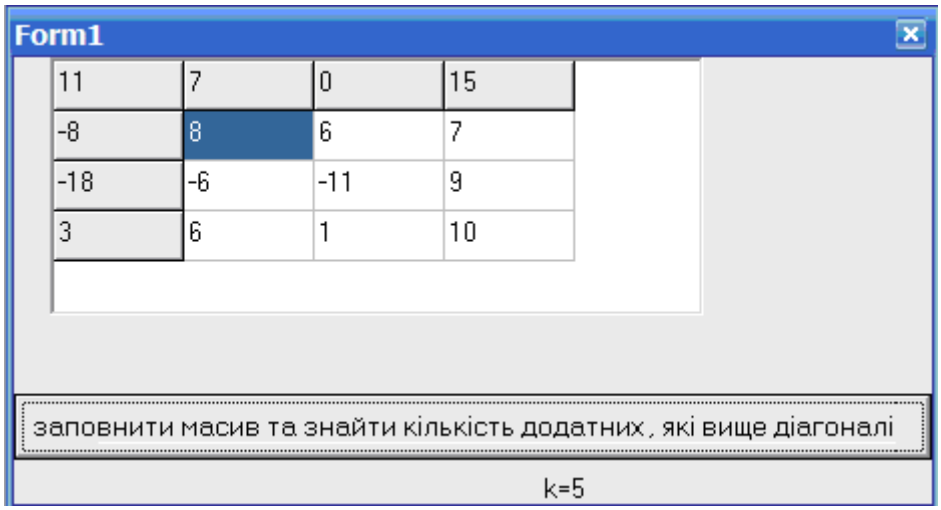


Рисунок 4.13 – Результат роботи коду

Приклад 3. Створити функцію для обчислення об'єму правильної піраміди зі стороною a і висотою h. Обчислити об'єм для значень a∈[3,01; 3,46]; Δa=0,05; h=12,4.

$$V = \frac{S_{осн} \cdot h}{3}; \qquad S_{осн} = \frac{a^2\sqrt{3}}{4}.$$

Програма в консольному режимі має вигляд:

```
#include<iostream.h>
#include<conio.h>
#include <math.h>
//---------------------------------------------------------------------------
float volume(float a,float h);                  //оголошення функції користувача
int main() {                                                    //головна функція
  float h=12.4, a=3.01;
  cout<< "  a        V"<<"\n";
  while (a<3.5)  {
    cout<<a<<"   "<<volume( a, h)<<"\n";                  //звернення до функції
    a+=0.05;
  }
  getch();
}
```

```
float volume(float a,float h) {                    //визначення функції користувача
  float s,v;
  s= a*a*sqrt(3)/4;
  v=s*h/3;
  return v;                                        //повернення в точку виклику
}
```

Результат виконання коду надано на рисунку 4.14.

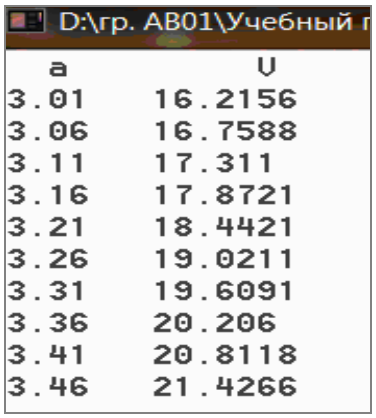


Рисунок 4.14 – Результат виконання коду

Запитання для самоконтролю

1. Що являє собою середовище Borland C++ Builder?
2. Наведіть приклади констант мови C++: цілих, з плаваючою точкою, символьних, строкових.
3. Перелічте основні типи даних мови C++.
4. Наведіть приклади оголошення та ініціалізації змінних.
5. Що являє собою консольний додаток в середовищі Borland C++ Builder?
6. Які існують можливості вводу-виводу в консольному режимі середовища Borland C++ Builder?
7. Поясніть особливості організації вводу-виводу в графічному середовищі Borland C++ Builder.
8. Розкажіть про стадії, які проходить програма в процесі виконання.
9. Яким чином здійснюється оголошення та ініціалізація масивів?
10. Які існують можливості організації розгалужень у програмі?

11. В яких випадках зручно користуватися оператором вибору switch?
12. Яким чином відбувається перетворення типів даних в мові С++?
13. Які існують можливості організації циклічних процесів в С++?
14. Як працює оператор циклу for?
15. В яких випадках зручно користуватися операторами циклу while, do…while?
16. Яким чином можна згенерувати масив випадкових чисел?
17. Що означає зона дії змінних?
18. Що являє собою динамічний розподіл пам'яті?
19. Що являє собою функція користувача?
20. Яким чином можна створити функцію користувача?

ЛІТЕРАТУРА

1. Шакин В. Н. Базовые средства программирования на Visual Basic в среде Visual Studio. Net / В.Н. Шакин. – М.: Форум, Инфра-М, 2015. – 304 с.
2. Швачич Г. Г. Інформатика та комп'ютерна техніка. Елементи об'єктно-орієнтованого програмування. Розділ «Реалізація концепції об'єктно-орієнтованого програмування в мові Visual Basic for Application»: Навчальний посібник / Г. Г. Швачич, О. В. Овсянніков. – Дніпропетровськ: НметАУ, 2006. – 52 с.
3. Дибкова Л.М. Інформатика і комп'ютерна техніка. 3-тє видання, доповнене / Л.М Дибкова. – Київ: Академвидав, 2014. – 464 с.
4. Пахомов Б. И. Самоучитель С/С++ и С++ Builder 2007 (+ DVD-ROM) / Б.И. Пахомов. – М.: БХВ-Петербург, 2013. – 672 с.
5. Перри Грег. Программирование на С для начинающих / Грег Перри, Дин Миллер. – М.: Эксмо, 2015. – 368 с.
6. Прата Стивен. Язык программирования С++. Лекции и упражнения / Стивен Прата. – М.: Вильямс, 2015. – 445 с.
7. Федоренко Ю. П. Алгоритмы и программы на С++ Builder (+ CD-ROM) / Ю.П. Федоренко. – М.: ДМК Пресс, 2010. – 544 с.
8. Ашарина И. В. Язык С++ и объектно-ориентированное программирование в С++. Лабораторный практикум. Учебное пособие / И.В. Ашарина, Ж.Ф. Крупская. – М.: Горячая линия – Телеком, 2015. – 232 с.

Навчальне видання

Швачич Геннадії Григорович
Гуляєва Олена Анатоліївна
Петречук Ліна Миколаївна

ОСНОВИ ПРОГРАМУВАННЯ

Навчальний посібник

Тем. план 2018, поз. 306

Підписано до друку Формат 60x84 1/16. Папір друк. Друк плоский.
Облік.-вид. арк. 5,59. Умов. друк. арк. 5,51. Тираж 100 пр. Замовлення № 43.

Національна металургійна академія України
49600, Дніпро, пр. Гагаріна, 4

Редакційно-видавничий відділ НМетАУ