

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

**О.М. Павловський,
С.Л. Лакоза,
С.С. Рупіч**

ПРОГРАМУВАННЯ КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Комп'ютерно-інтегровані системи та технології в
приладобудуванні»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2022

Рецензенти: *Маркін М.О.* - канд. техн. наук, доцент, доцент кафедри інформаційно-вимірювальних технологій, КПІ ім. Ігоря Сікорського
Головач С.В. - канд. техн. наук, головний конструктор напрямку АТ «Елміз»

Відповідальний редактор: *Цибульник С.О.* - канд. техн. наук, доцент, доцент кафедри комп'ютерно-інтегрованих оптичних та навігаційних систем КПІ ім. Ігоря Сікорського

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 4 від 07.04.2022 р.)
за поданням Вченої ради Приладобудівного факультету
(протокол № 2/22 від 28.02.2022 р.)*

Навчальний посібник містить теоретичні відомості та практичні рекомендації до виконання комп'ютерних практикумів з дисципліни «Програмування».

У процесі виконання комп'ютерних практикумів будуть розглянуті питання використання основних конструкцій мови C++, створення консольних додатків та додатків із графічним інтерфейсом, із поступовим ускладненням задач, що буде спонукати здобувача до подальшого, більш глибокого опанування основних моментів мови програмування.

Навчальний посібник призначений для здобувачів ступеня бакалавра за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології», буде також корисним студентам споріднених технічних спеціальностей.

Реєстр. № НП 21/22-382. Обсяг 4,54 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056

<https://kpi.ua>

Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

© О.М. Павловський, С.Л. Лакоза, С.С. Рупіч, 2022

© КПІ ім. Ігоря Сікорського, 2022

ЗМІСТ

ВСТУП	4
Комп'ютерний практикум № 1. Створення найпростішого додатку.....	5
Комп'ютерний практикум № 2. Дії з числами.....	21
Комп'ютерний практикум № 3. Програмування лінійних алгоритмів.....	29
Комп'ютерний практикум № 4. Створення додатка з декількома формами.....	40
Комп'ютерний практикум №5. Знайомство з інструкцією вибору.....	47
Комп'ютерний практикум №6. Програмування розгалужених алгоритмів....	54
Комп'ютерний практикум №7. Циклічні алгоритми з фіксованою кількістю повторень.....	60
Комп'ютерний практикум №8. Цикли із невизначеною кількістю ітерацій.....	65
Комп'ютерний практикум №9. Масиви. Пошук даних в одновимірних масивах.....	73
Комп'ютерний практикум №10. Пошук даних в багатовимірних масивах....	89
Список рекомендованих джерел.....	100

ВСТУП

У сучасному світі за останнє десятиріччя використання комп'ютерів перейшло на істотно новий рівень, пропорційно підвищилась і кількість використовуваного програмного забезпечення, причому у будь-якій сфері людської діяльності та повсякденному житті. Якісно створене програмне забезпечення прискорює, а іноді і повністю автоматизує виконання будь-яких задач. Відповідно, для створення таких програм, програміст повинен досконало володіти знанням синтаксису відповідної мови програмування, основними прийомами розробки інтерфейсу, архітектури, функціоналу програмного забезпечення.

Таким чином, метою даного посібника буде вирішення задачі оволодіння синтаксисом мови C++ на відповідному рівні, її основними конструкціями, операторами та стандартними підходами до створення програм. З урахуванням того, що мова програмування є своєрідним інструментом, її опанування дозволить студенту у подальшому вирішувати прикладні задачі в межах спеціальності під час навчання та, у подальшому, під час трудової діяльності.

У розділах посібника, які сформовані у вигляді окремих комп'ютерних практикумів, будуть розглянуті питання використання основних конструкцій мови C++, створення консольних додатків та додатків із графічним інтерфейсом, із поступовим ускладненням задач, що буде спонукати здобувача для подальшого, більш глибокого опанування основних моментів мови програмування.

У посібнику пропонується реалізовувати приклади та завдання у середовищах Borland C++ Builder, Embarcadero RAD Studio або MS Visual Studio, що на сьогодні є найбільш розповсюдженими середовищами програмування мовою C++.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 1

СТВОРЕННЯ НАЙПРОСТІШОГО ДОДАТКУ

Мета роботи: створити простий C/C++ додаток, що дозволяє виводити текстову інформацію у консоль або форму.

1.1. Теоретичні відомості

Програмування представляє собою процес визначення послідовності інструкцій, які повинен виконати комп'ютер для вирішення певної задачі. Для реалізації таких інструкцій зазвичай використовується мова програмування. У даних комп'ютерних практикумах буде використовуватися мова програмування C++.

Кожна мова програмування має певні команди або програмні оператори, які є зрозумілими для людей. Вони використовуються для написання програм. Тобто, в загальному та широкому розумінні, програми – це текстові файли з набором операторів, які обробляються в заданому програмістом порядку та у відповідності до встановлених правил. Для того щоб комп'ютер мав змогу розрізнити ці оператори, використовується спеціальна програма – компілятор, який перетворює оператори із формату, зрозумілого для людини, в набір одиниць та нулів, які розуміє комп'ютер. Тобто, компілятор – це своєрідний перекладач, що слугує налагодженню комунікації між людиною та комп'ютером. При порушенні одного чи декількох правил програмування на C++ компілятор виведе повідомлення на екран про семантичні помилки. У разі виникнення таких помилок поточну програму потрібно відредагувати, щоб їх виправити, та запустити процес компіляції повторно.

У часи, коли програмування ще не набуло широкого розвитку, більшість програм писалися в текстових редакторах. Ця можливість залишається і сьогодні. Однак, розвиток програмування, створення документації, додаткових модулів і бібліотек, попит на створення складних програм і додатків – усі ці фактори все більше ускладнювали процес розробки. Налаштування звичайного текстового редактора вимагає все

більше часу, а при виникненні помилок, потребує значних ресурсів для їх знаходження та виправлення, що викликає дискомфорт під час роботи. Оскільки людство намагається спрощувати певні рутинні процеси, тому розробники об'єднали свої зусилля та створили багато корисних інструментів. Одним із таких інструментів є середовища розробки, так звані IDE (integrated development environment – інтегроване середовище розробки), які значно спрощують та полегшують процеси написання коду та створення програм.

IDE – це не просто текстовий редактор. У той час як сучасні текстові редактори для коду, наприклад Sublime, пропонують безліч зручних функцій, таких як: підсвічування синтаксису, налаштування інтерфейсу і розширені засоби навігації. Але вони надають можливість лише писати код. Для створення функціонуючих програм, як мінімум, потрібен компілятор і інструмент налагодження. IDE включає в себе ці компоненти, як і ряд інших. Деякі з них поставляються з додатковими інструментами для автоматизації, тестування та візуалізації процесу розробки. Термін «інтегроване середовище розробки» означає, що надається все необхідне для перетворення коду в функціонуючі програми.

На сьогоднішній день є багато середовищ програмування на мові C++. Найбільш відомі та поширені: Visual Studio Code, NetBeans, Code::Blocks, Embarcadero RAD Studio, Borland C++ Builder та інші. Для виконання комп'ютерних практикумів, необхідно інсталиювати (встановити) будь-яку з вищезазначених IDE.

1.1.1. Структура програми

Перш ніж починати програмувати, потрібно зрозуміти структуру програм на мові програмування C++.

Структура програми – це розмітка робочої області (області коду) з метою чіткого визначення основних блоків програм і синтаксису.

У залежності від IDE, структура програми може дещо відрізнятися. У подальшому основну увагу буде зосереджено на роботі в IDE *Embarcadero RAD Studio* та *Borland C++ Builder*.

У загальному вигляді, структура програми C++ складається з наступних частин:

- директив препроцесора (або заголовочних файлів);
- указівок компілятору;
- оголошення змінних і/або констант;
- оголошення функцій;
- головної функції `main()`, з якої починається виконання програми;
- визначення функцій.

Наприклад, загальну структуру програми можна описати наступним чином (табл. 1.1):

Таблиця 1.1 – Загальна структура програми

<code>#include <ім'я бібліотеки 1></code> <code>#include <ім'я бібліотеки 2></code>	Заголовочні файли (підключення бібліотек)
<code>// прототипи функцій</code>	Оголошення функцій
<code>// глобальні ідентифікатори</code> (типи, змінні і т.п.)	Оголошення глобальних ідентифікаторів
<code>int main()</code> { <code>// опис змінних</code> <code>// розділ операторів</code> }	Головна функція програми
<code>// реалізація функцій</code>	Реалізація оголошених функцій

1.1.2. Директиви препроцесора

Директива препроцесора – це інструкція, що включає в текст програми файл, що містить опис багатьох функцій, які надають змогу правильно компілювати (запускати) програму.

Важливо:

- усі директиви препроцесора починаються зі знаку `#`;
- після директиви препроцесору крапка з комою **не ставиться**.

Директива `#include` надає можливість включати в текст програми вказаний файл. Ім'я файлу може бути вказано двома способами:

1) `#include <some_file.h>`

2) `#include "my_file.h"`

Якщо файл є стандартною бібліотекою і знаходиться в теці компілятора, то він прописується між кутовими дужками `< >`.

Якщо файл знаходиться в поточному каталозі проекту, він вказується між подвійними лапками `" "`.

Синтаксис для підключення заголовочних файлів такий:

`#include <ім'я заголовочного файлу>`

Проте, часто необхідно використовувати функціонал з більш старих бібліотек. Оскільки C++ успадкувала багато інструментів з мови програмування C, то більш старі заголовочні файли підключаються наступним чином:

`#include <ім'я заголовочного файлу.h>`

Відмінність полягає в тому, що вкінці, після імені файлу, обов'язково вказується розширення `.h`.

Стандартна бібліотека – це колекція класів і функцій, що написані на базовій мові.

Основні заголовочні файли:

- `iostream` – підключає потоки вводу/виводу;
- `fstream` – підключає файлові потоки;
- `sstream` – підключає строкові потоки;
- `math.h` (`cmath`) – підключає математичні функції.

Наприклад, директива `#include <iostream>` – використовується для підключення зовнішнього файлу в програму, у даному випадку це файл `iostream`, що відповідає за підтримку системи вводу-виводу.

1.1.3. Простір імен

Варто зазначити, що до директив також відносять оператор `using`, що відкриває доступ до простору імен (англ. `namespace`).

Простори імен призначені для локалізації імен ідентифікаторів та використовуються з метою запобігання їх конфліктів.

Середовище програмування C++ працює з великою кількістю змінних, функцій і класів. Раніше всі імена перебували в глобальному просторі та

дуже часто конфліктували між собою. Мається на увазі, використання однакових імен з різних бібліотек, тобто, коли програма використовує декілька сторонніх бібліотек одночасно, кожна з яких містить визначені елементи з однаковими іменами. Так, наприклад, функція $\sin(x)$ зустрічається в 2 різних бібліотеках і компілятор не розуміє, з якого файлу необхідно взяти цю функцію для підрахунку. Такі конфлікти найбільш часто виникають із-за імен класів. Введення ключового слова *namespace* надало змогу вирішити ці проблеми.

Оскільки простір імен надає можливість локалізувати область видимості елементів, що оголошені всередині нього, одне і теж ім'я, але використане в різних контекстах, більше не викликає конфліктів. Найбільшу користь це нововведення принесло стандартній бібліотеці мови C++. Раніше вся стандартна бібліотека мови C++ знаходилася в глобальному просторі імен (тобто, це простір, який є єдиний для всієї програми). Тепер стандартна бібліотека визначена всередині свого власного простору імен *std*, що значно зменшує ймовірність виникнення конфліктів.

Програміст може створювати свої власні простори імен і самостійно локалізувати імена, які можуть викликати конфлікти. Це особливо важливо при розробці класів і бібліотек функцій.

Отже, простір імен (*namespace*) – це середовище, що створене для логічного групування унікальних імен, яке необхідне для запобігання конфліктів імен ідентифікаторів.

У просторі імен *std* знаходиться вся стандартна бібліотека – це бібліотека, що включає набір класів і функцій, які написані на базовій мові і є частиною стандарту C++ ISO. Вона включає основні стандартні файли заголовків для контейнерів, потоків вводу\виводу, рядкових, числових бібліотек та інші.

Синтаксис для підключення простору імен *std* такий:

using namespace std;

Без підключення простору імен *std*, за замовчуванням використовується стандартний простір

using namespace standart;

Різниця у використанні просторів імен представлена у таблиці 1.2.

Таблиця 1.2 – Приклад використання операторів у просторах імен `std` і `standart`

Std	standart
<code>#include <iostream></code> <code>using namespace std;</code>	<code>#include <iostream></code>
<code>int main()</code> <code>{</code> <code>int a, b, c;</code>	<code>int main()</code> <code>{</code> <code>int a, b, c;</code>
<code>cout << "Запишіть значення:</code> <code>\n";</code> <code>cin >> a >> b >> c;</code>	<code>std:: cout << "Запишіть</code> <code>значення: \n";</code> <code>std:: cin >> a >> b >> c;</code>
<code>system("pause");</code> <code>return 0;</code> <code>}</code>	<code>system("pause");</code> <code>return 0;</code> <code>}</code>

Можна побачити, що певні елементи програми в стандартному просторі імен викликаються у вигляді **`std:: ім'я об'єкта`**. У процесі розробки, може виникнути ситуація, коли в одному рядку коду буде використано декілька об'єктів зі стандартної бібліотеки, що призведе до погіршення читабельності коду та значному розширенню команди. Тому, на практиці, простір імен стандартної бібліотеки **`std`** підключається майже завжди.

Підсумовуючи вищезазначені викладки, структура програми на C++ виглядає наступним чином (рис. 1.1):

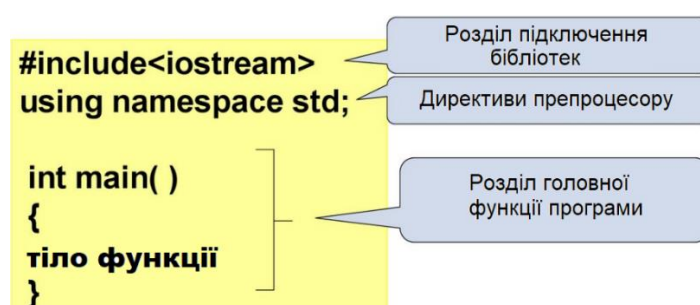


Рис. 1.1 – Загальна структура програми C++

Варто зазначити, що у середовищі *Borland C++ Builder 6* підключати простір імен не потрібно.

1.1.4. Функція `main()`

Написання програми на мові C++ починається зі спеціальної стартової функції *`main()`*. У момент запуску програми, керування передається цій

функції. Одне з основних правил мови програмування C++ – програма обов’язково повинна містити функцію *main* в одному зі своїх модулів.

Стандарт передбачає два формати функції:

- без параметрів:

```
int main()  
{  
/* набір інструкцій */  
}
```

- з двома параметрами:

```
int main(int argc, _TCHAR* argv[])  
{  
/* набір інструкцій */  
}
```

Якщо програму запускати через командний рядок, то існує можливість передавати будь-яку інформацію цій програмі.

Параметр *argc* типу *int* містить кількість параметрів, що передаються в функцію *main()*. Крім того, *argc* завжди не менше 1, навіть коли функції *main()* не передається жодної інформації. У такому випадку першим параметром вважається ім’я програми.

Параметр *argv[]* являє собою масив вказівників на рядки.

Функція *main()* може повертати певне значення або не повертати нічого.

Якщо функція нічого не повертає, то вона повинна мати тип даних *void* при її оголошенні та визначенні (такі функції, в інших мовах програмування, іноді називають процедурами). На рис. 1.2 наведені приклади головної функції *main()* різного типу.

<pre>int main() { ... }</pre>	або	<pre>void main() { ... }</pre>
---	-----	--

Рис. 1.2 – Функція *main()* з різним типом даних

Структура головної функції наведена на рис. 1.3:

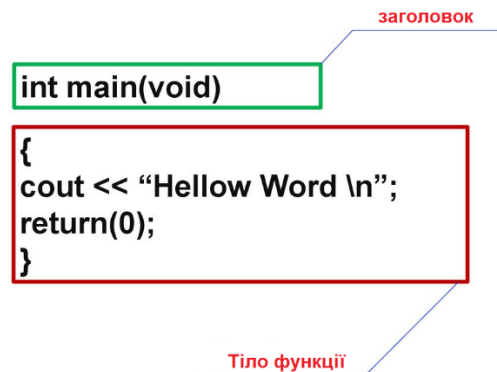


Рис. 1.3 – Структура головної функції

Розглянемо рядок *int main(void)*:

- *int* – це тип значення, яке повертає функція. У даному випадку це *int*. Тобто, коли функція *main()* завершить виконання своєї роботи, вона повинна повернути в програму, що її викликала, яесь ціле значення.

Якщо не потрібно, щоб програма повертала яесь значення, тоді можна використовувати тип даних *void*. Наприклад, якщо функція *main* не повинна нічого повертати, то її заголовок виглядав би наступним чином: ***void main (void)***. **Увага!** Для *Embarcadero RAD Studio* функція ***_tmain()*** завжди має повертати цілочисельне значення.

- ім'я функції ***main()***. Головна функція завжди має ім'я ***main()*** (***_tmain*** для **Rad Studio**). Програма може містити безліч різних функцій з різними іменами. Навіть такі, як ***main1()*** або ***main2()***. Але наявність головної функції ***main()*** є обов'язковим, щоб програма вдало скомпілювалася.

- останньою частиною запису, в круглих дужках, є тип даних та кількість аргументів (параметрів) функції. У даному прикладі записано тип даних *void*, що означає, що функція не приймає ніяких аргументів. Тобто, (*void*) – це перелік аргументів функції. Ключове слово *void* вказує, що у даній функції аргументи відсутні.

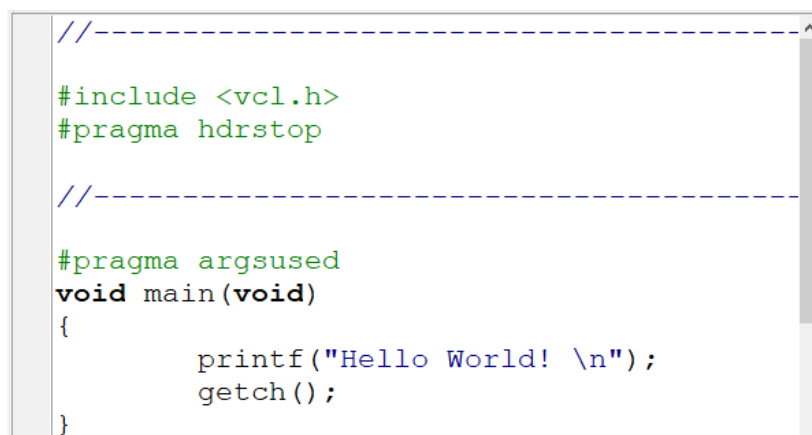
1.2.1 Завдання 1. Консольний додаток.

Для початку потрібно познайомитися з мовою програмування та, класично, “привітатися зі світом” на мові програмування C/C++. Код першого додатка повинен відображати текст "Привіт світ!" і

тримати зображення, щоб воно "не втекло", поки користувач розглядає те, що з'явилося на екрані. Перед тим як почати, необхідно створити новий проект скориставшись меню File-> New-> Other...Console Wizard (Borland C++ Builder), або File-> New->Console Application - C++ Builder (Embarcadero RAD Studio). Також необхідно зберегти створений проект використовуючи меню File-> Save All.

Тепер, у тілі головної функції, як показано на рис. 1.4 та рис. 1.5 необхідно написати наступний код:


```
printf("Hello world!\n");  
getch();
```



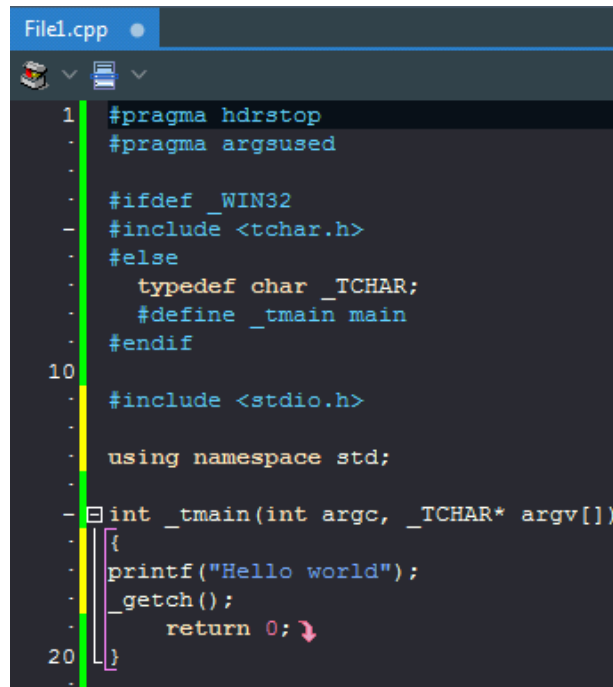
```
//-----  
  
#include <vcl.h>  
#pragma hdrstop  
  
//-----  
  
#pragma argsused  
void main(void)  
{  
    printf("Hello World! \n");  
    getch();  
}
```

Рис. 1.4 – Робоче вікно з кодом в середовищі Borland C++ Builder 6

Слід зазначити, що для двох середовищ програмування є певні відмінності у написанні коду, а саме: в *Embarcadero RAD Studio* було підключено простір імен *std*; функція *getch()* з *Borland C++ Builder* вказується як *_getch()* в *Embarcadero RAD Studio*.

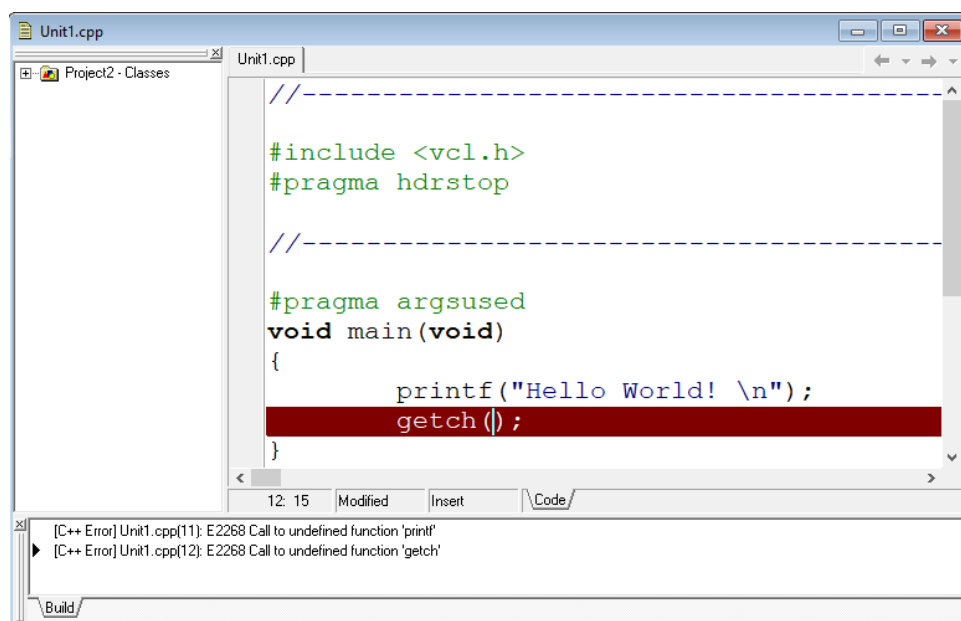
Для того, щоб додаток працював, він повинен бути скомпільований, тобто переведений з коду мови C/C++ у машинний код. Для цього необхідно запустити програму компілятора. Це робиться за допомогою "гарячої" клавіші <F9>, іконки  або за допомогою пункту Головне меню - > Run -> Run. Якщо це зробити, то буде отримано реакцію середовища, яка зображена на рис. 1.6. На даному рисунку видно, що компіляція зазнала невдачі: у нижній частині вікна було виведено повідомлення про дві помилки: "Викликано невідому функцію". Якщо двічі натиснути на кожному рядку з

інформацією про помилку, то в функції **main()** буде виділено той рядок, в якому було виявлено дану помилку.



```
File1.cpp
1  #pragma hdrstop
   #pragma argsused
   .
   .
   #ifdef _WIN32
   #include <tchar.h>
   #else
   typedef char _TCHAR;
   #define _tmain main
   #endif
10 #include <stdio.h>
   .
   using namespace std;
   .
   int _tmain(int argc, _TCHAR* argv[])
   {
   printf("Hello world");
   _getch();
   return 0;
   }
```

Рис. 1.5 – Робоче вікно з кодом в середовищі Embarcadero RAD Studio



```
Unit1.cpp
Project2 - Classes
Unit1.cpp
//-----
#include <vcl.h>
#pragma hdrstop
//-----
#pragma argsused
void main(void)
{
printf("Hello World! \n");
getch ();
}
12: 15 Modified Insert \Code/
[C++ Error] Unit1.cpp(11): E2268 Call to undefined function 'printf'
[C++ Error] Unit1.cpp(12): E2268 Call to undefined function 'getch'
Build/
```

Рис. 1.6 – Вікно коду із помилкою

Якщо подивитись довідку на функцію `printf()` (довідка в C++Builder, або мережа інтернет), то можна побачити, що дана функція описується заголовочним файлом `stdio.h`. Отже, для того, щоб середовище розуміло функцію `printf()` над функцією `main()` необхідно прописати директиву препроцесора: `#include <stdio.h>`.

Директива `#include` пропонує компілятору підключити до проекту файл, ім'я якого вказується після директиви. Файл береться в подвійні лапки або в `< >` у залежності від фізичної адреси файлу. Стандартні заголовочні файли завжди беруться у трикутні дужки. Таким же чином можна знайти, що для коректного використання невідомої функції `getch()` до проекту слід підключити рядок `#include <conio.h>`. Після цього текст програми буде виглядати, як показано на рис. 1.7.

```
Unit1.cpp
//-----
#include <vcl.h>
#pragma hdrstop
#include <conio.h>
#include <stdio.h>

//-----

#pragma argsused
void main(void)
{
    printf("Hello World! \n");
    getch();
}
```

Рис. 1.7 – Текст програми після підключення необхідних бібліотек

Після повторної компіляції, повинно відобразитись вікно консолі із наступним текстом (рис. 1.8):

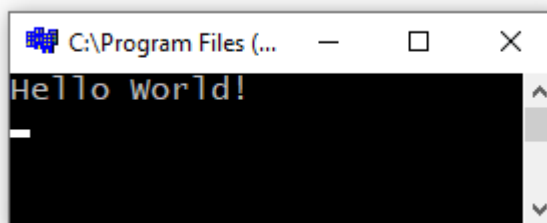


Рис. 1.8 – Результат роботи першої програми

Це означає, що програма успішно компілюється і виконується. Якщо тепер натиснути будь-яку клавішу, програма завершиться, і знову буде видно її код.

Будь-яка C/C++ програма будується як сукупність елементів, які називаються функціями, умовно, блоків програмного коду, які виконують

певні дії. Імена цих блоків коду, побудованих за спеціальними правилами, задає або програміст, якщо він сам їх конструює, або імена вже задані в бібліотеці стандартних функцій. Ім'я головної функції, з якої і починається виконання програми, задано в середовищі програмування. Це ім'я - `main()` (`_tmain` для Rad Studio). У процесі виконання програми сама функція `main()` обмінюється даними з іншими функціями і використовує їх результати роботи. Обмін даними між функціями відбувається через параметри функцій, які вказуються в круглих дужках, розташованих слідом за ім'ям функції. Функція може і не мати параметрів, але круглі дужки після імені завжди повинні бути присутніми: по ним компілятор дізнається, що перед ним функція, а не інша конструкція мови.

У наведеному вище прикладі дві функції, використані в головній функції `main()`: `printf()` та `getch()`.

Функція `printf()`, як аргумент, приймає рядок взятих у подвійні лапки символів. Серед символів цього рядка є спеціальний символ, або по іншому, керуюча символна послідовність, записана як: `\n`. Один з перших 32-х символів таблиці кодування символів ASCII. Керуючі символи не мають екранного відображення і використовуються для управління процесами. У даному випадку символ `\n` служить для установки покажчика зображення символів на екрані (екранного курсору) у першу позицію, тобто в початок наступного рядка. Це означає, що коли працює функція `printf()`, символи рядка по одному записуються в певний буфер до тих пір, поки не зустрінеться послідовність символів `\n`. Як тільки послідовність `\n` прочитана, вміст буфера тут же передається на пристрій виводу (у даному випадку – на екран).

Функція `getch()` – це функція, що очікує введення одного символу з клавіатури: вона чекає натискання будь-якої клавіші. Завдяки цій функції результат виконання програми затримується на екрані до тих пір, поки не буде натиснено будь-який символ на клавіатурі. Якби в коді не було функції `getch()`, то після виконання `printf()` програма дійшла б до кінця тіла функції `main()` і завершила б свою роботу. У результаті вікно консолі, в якому

вивелося повідомлення Hello world !, закрилося, і користувач не встиг би побачити результат роботи програми.

Слід зазначити, що основне призначення функції getch() – очікувати введення символу за допомогою клавіатури і передавати їх символьним змінним. Проте для наведеного прикладу було використано побічну властивість даної функції – чекати введення з клавіатури і, тим самим, не дати програмі завершитися, щоб користувач зміг побачити результат її попередньої роботи.

1.2.2. Завдання 2. Знайомство з формами

Для того щоб створити перший додаток із графічним інтерфейсом, необхідно скористатися меню File вибрати New -> Application(Windows VCL Application). З'явиться порожнє вікно форми. Наступні ілюстрації відносяться до середовища *Borland C++ Builder 6*, однак, всі вони справедливі для *Embarcadero RAD Studio*, лише мають деякі відмінності у інтерфейсі.

Далі із палітри стандартних інструментів



, обрати елемент Label і розмістити на формі. В *Embarcadero RAD Studio* цей інструмент знаходиться на палітрі «Palette» праворуч основного вікна.

Далі обравши елемент Label1, що з'явився на формі, необхідно змінити його властивості, що впливають на його зовнішній вигляд. Властивості можуть бути змінені або програмно, або за допомогою інспектора, вигляд якого показаний на рис. 1.9.

Так, обравши властивість Caption можна змінювати напис, що буде відображено в додатку на формі.

Змініть різні властивості об'єкта Label1 і дослідіть функції цих властивостей.

Загальний вигляд додатку повинен бути таким, як зображено на рис. 1.10.

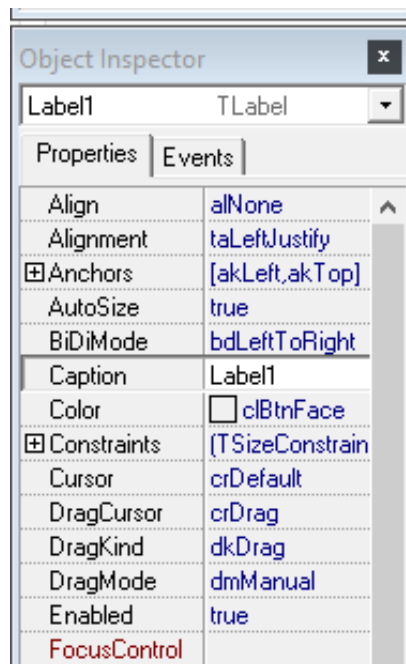


Рис. 1.9 – Інспектор об'єктів, і вкладка Properties (Властивості)

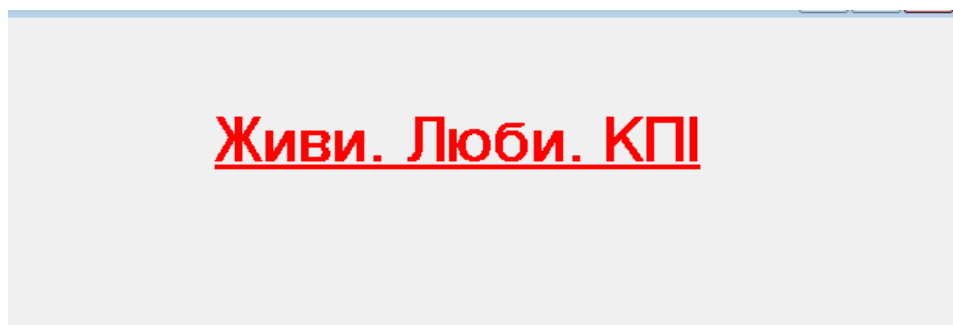


Рис. 1.10 – Загальний вигляд першого додатку у формі

Тепер необхідно додати код. Розмістіть на вільному місці форми кнопку Button із палітри стандартних інструментів.

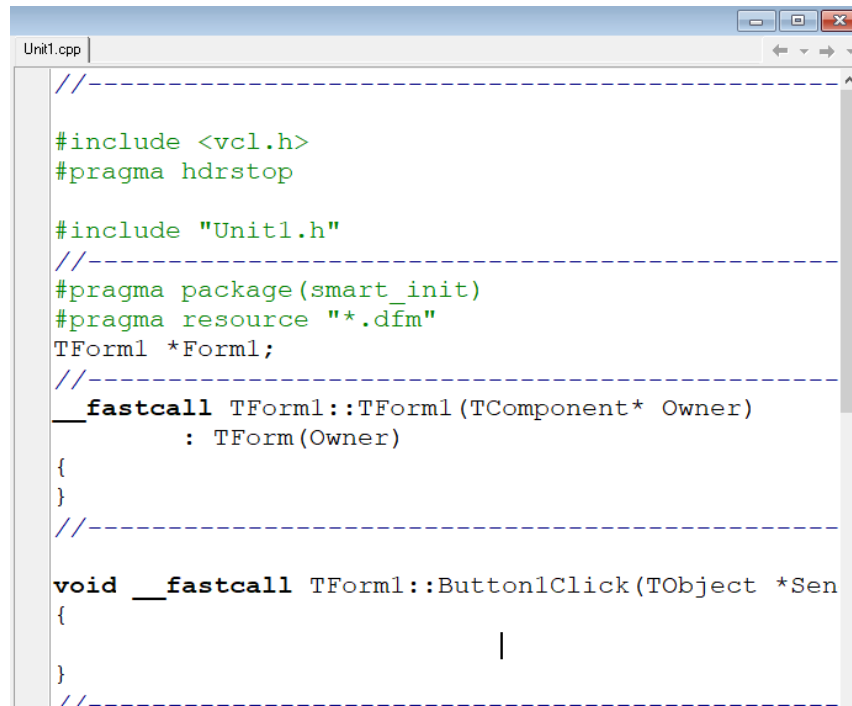
Змініть напис Button1 на Ок, використовуючи властивість Caption.

Зробивши подвійний клік по кнопці, відкриється вікно коду із створеним обробником подій, загальний вигляд якого показано на рис. 1.11.

У блоці коду (між фігурними дужками) обробника подій TForm1::Button1Click необхідно написати наступний код:

```
Label1->Caption="Героям Слава!!!";
Label1->Font->Color=clYellow; ,
```

де Label1 – об’єкт із яким буде проходити взаємодія, Caption – властивість об’єкта, яка містить текст для відображення. Символи -> дозволяють звертатись до властивостей будь-яких об’єктів.

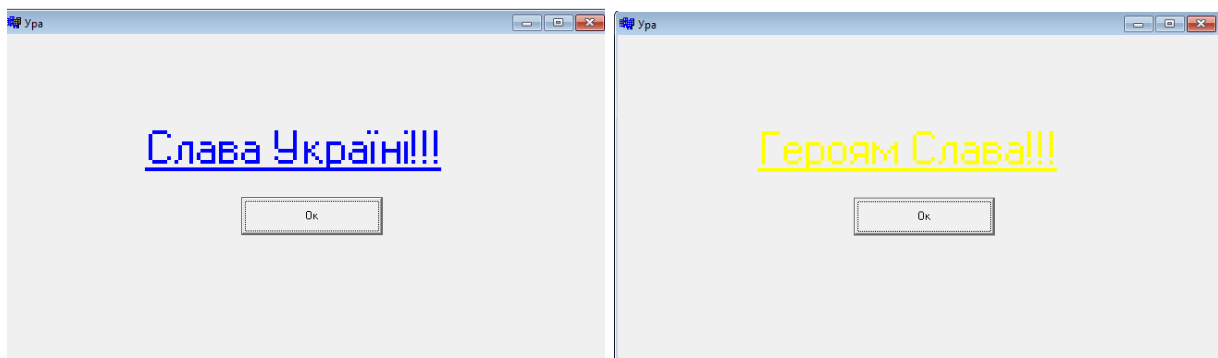


```
//-----  
#include <vcl.h>  
#pragma hdrstop  
#include "Unit1.h"  
//-----  
#pragma package(smart_init)  
#pragma resource "*.dfm"  
TForm1 *Form1;  
//-----  
__fastcall TForm1::TForm1(TComponent* Owner)  
    : TForm(Owner)  
{  
}  
//-----  
void __fastcall TForm1::Button1Click(TObject *Sender)  
{  
    |  
}  
//-----
```

Рис. 1.11 – Вікно коду після натискання кнопки Button1

Оскільки властивість Color вкладена у властивість Font, то необхідно звернутись спочатку до ->Font, а вже потім до ->Color.

Змінюючи властивості об’єктів, необхідно отримати додаток із таким функціоналом (рис. 1.12):



а)

б)

Рис. 1.12 – Готовий додаток: а) Перед натисканням на кнопку;
б) Після натискання на кнопку

1.3. Зміст звіту

1. Код створених програм (перенести за допомогою Print Screen або скопіювавши). До коду бажано додавати коментарі, що пояснюють основні моменти і особливості виконання програми.
2. Фото із результатами виконання як в консолі, так і у формі.
3. Загальні висновки по роботі.

1.4. Контрольні запитання

1. З чого починається виконання програми в C++?
2. Опишіть загальну структуру програми.
3. Що таке директива препроцесора?
4. Як підключити директиву препроцесора?
5. Чим відрізняється підключення заголовочних файлів з використанням <бібліотека 1> від "бібліотека 2"?
6. Навіщо потрібен простір імен?
7. Який заголовочний файл підключає потік вводу-виводу?
8. Як позначається блок коду?
9. Як у Application проекті звернутися до властивості об'єкту?
10. Який символ потрібно використовувати, щоб компілятор C++ зрозумів, що в коді використовується рядок тексту?
11. Як позначається однорядковий та багаторядковий коментар в C++?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 2

ДІЇ З ЧИСЛАМИ

Мета роботи: створити додаток, що дозволяє працювати із математичними операторами використовуючи діалоговий режим введення та виведення інформації.

2.1. Теоретичні відомості

Потокове введення-виведення в C/C++ виконується за допомогою підключення стандартних бібліотек. В C++, як і в C, немає вбудованих засобів введення-виведення. У мові C для цього використовується бібліотека *stdio.h*. В C++ розроблена нова бібліотека введення-виведення *iostream*, що використовує концепцію об'єктно-орієнтованого програмування. Синтаксис для підключення бібліотеки:

- 1) **#include <iostream.h>** – для *Borland C++ Builder*.
- 2) **#include <iostream>** – для *Embarcadero RAD Studio* або *Visual Studio*.

Бібліотека *iostream* налічує значну кількість операторів для роботи із поточковим вводом/виводом, проте ми будемо використовувати наступні:

- **cin** – стандартний вхідний потік (введення інформації);
- **cout** – стандартний вихідний потік (виведення інформації).

Для виконання операцій вводу-виводу використовуються наступні оператори:

- **>>** – отримати із вхідного потоку;
- **<<** – перемістити в вихідний потік.

2.2. Завдання 1. Додавання двох цілих чисел

Необхідно створити додаток, що буде додавати два цілих числа введених з клавіатури і виводити результат додавання.

Порядок виконання

1. Створіть і збережіть новий консольний проект.
2. Підключіть заголовочний файл *iostream*, без якого компілятор не зрозуміє функції потоків операторів **cin** та **cout**.

3. Оголосить дві змінні типу **int**. Наприклад `int integer1, integer2;` Це означає, що ці змінні завжди будуть зберігати дані цілочисельного типу, тобто цілі числа (наприклад, 7, -11, 0, 100500).

4. Запишіть перший оператор:

```
cout << "Enter First Number\n"; // Запит на введення даних
```

Цей оператор можна прочитати наступним чином: «**cout** отримує символну послідовність "Enter First number\n"». Він друкує на екрані повідомлення, що стоїть між подвійними лапками та переносить курсор на початок наступного рядка. Варто зауважити, що будь-яке текстове повідомлення завжди включають у подвійні лапки. Це повідомлення називається запитом, тому, що воно пропонує користувачу виконати певну дію.

5. Далі запишіть оператор:

```
cin >> integer1;
```

У цьому рядку використовуються оператор вхідного потоку **cin** та операцію отримати із вхідного потоку `>>`, щоб отримати значення, яке ввів користувач. Оператор **cin** отримує інформацію, що була введена в стандартний потік вводу, яким зазвичай є клавіатура. Тобто, цей запис можна прочитати наступним чином: «**cin** призначає значення першого цілого числа змінній `integer1`».

Оператори потоків **cout** та **cin** виконують взаємодію між користувачем і комп'ютером. Оскільки ця взаємодія нагадує діалог, часто кажуть що це діалоговий розрахунок або інтерактивний розрахунок.

6. Повторіть дії п.п. 3-5 для змінної `integer2`.

7. Для підсумовування значень, що збережені в змінних напишемо:

```
int sum = integer1 + integer2; // присвоєння значення суми змінній sum
```

Дана команда розраховує суму змінних **integer1** та **integer2** і присвоює результат дії новій змінній **sum**. Для цього використовується операція присвоєння `=`. Оператор присвоєння використовується в більшості розрахунків та є одним з найбільш вживаних у мові програмування C++.

8. Для виводу інформації запишемо наступний код:

```
cout << "The sum is " << sum << endl; // друк суми
```

Ця команда друкує символний рядок «The sum is », після чого виводить чисельне значення змінної `sum`. Далі використовується оператор `endl`, що є аббревіатурою словосполучення «end line» – кінець рядка. Оператор `endl` ще називають *маніпулятором потоку*. Він виводить символ нового рядка, а далі очищає буфер виводу.

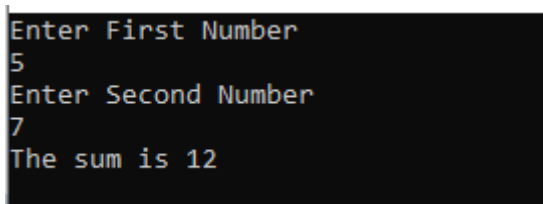
Варто зауважити, що записана команда виводить множину значень різних типів. Тобто, операція помістити в потік знає, як виводити кожний елемент даних. Багаторазове використання операції помістити в потік (`<<`) в одному операторі називається *зчепленою операцією помістити в потік*. Таким чином, не обов'язково мати множину операцій виводу для друку множини фрагментів даних, змінних і т.п., що є зручним і призводить до скорочення написання коду.

В операторах виводу можна також виконувати обчислення. Наприклад, дві попередні команди можна об'єднати в один оператор:

```
cout << "The sum is " << integer1 + integer2 << endl;
```

Таким чином не потрібно оголошувати змінну `sum`.

Після компіляції результат виконання програми виглядає наступним чином (рис. 2.1):



```
Enter First Number
5
Enter Second Number
7
The sum is 12
```

Рис. 2.1 – Результат виконання програми

Треба зазначити, що за замовчуванням вивід у консоль не підтримує кириличні літери, тому, щоб надрукувати діалогові фрази українською, у головну функцію необхідно додати рядки:

```
SetConsoleOutputCP(1251);
```

```
SetConsoleCP(1251);
```

2.3. Завдання 2. Дії із графічним інтерфейсом

Створіть додаток із графічним інтерфейсом, який при натисканні кнопки множить два числа, які введені користувачем, та показує результат виконання множення.

При побудові цього додатку використовуйте нові типи компонентів – вікна редагування **LabeledEdit**. Для різноманітності, можна виводити результат не в мітку **Label**, а в панель **Panel**, щоб випробувати новий компонент.

Спершу, створіть новий проект із графічним інтерфейсом, для цього виконайте **File-> New->Console Application - C++ Builder**, як показано у попередньому комп'ютерному практикумі.

На форму із вкладки інструментів **Additional** розмістіть два вікна редагування **LabeledEdit**, які є зручною комбінацією об'єктів **Label** та **Edit**, поєднуючи в собі властивості обох цих об'єктів. Із вкладки інструментів **Standard**, перенесіть один об'єкт **Label** для напису, один об'єкт панелі **Panel** та один об'єкт кнопки **Button**. Після виконання дій, загальний вигляд додатку має бути наступним (рис. 2.2):

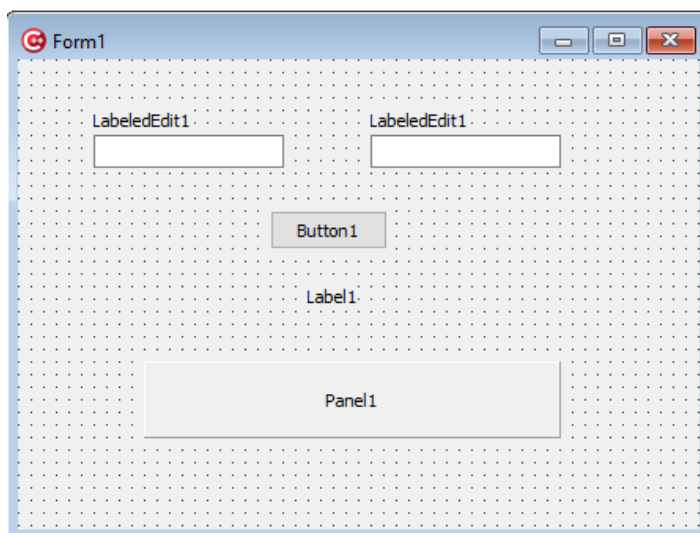


Рис. 2.2 – Вигляд форми після перенесення на нього об'єктів

Змініть надписи на мітках компонентів **LabeledEdit** на щось змістовне. Наприклад, надписи «Число 1», «Число 2», «Результат» (або «Ширина», «Висота», «Площа», тощо). Для цього потрібно натиснути на символі «+» у

властивостях **EditLabel** цих компонентів і змінити напис у властивості **Caption** (знайти у відкритому списку властивостей). Можна також задати для написів тип шрифту **bold**.

Замініть напис у властивості **Caption** кнопки, наприклад, на «Розрахунок». У об'єкті **Panel** варто очистити властивість **Caption**. У властивості **Text** (текст) вікон редагування задайте «1» – початкове значення тексту.

Спробуйте варіювати такими властивостями панелі, як **BevelInner** і **BevelOuter**, які визначають вигляд (втоплений – **bvLowered** або випуклий **bvRaised**) основного поля та рамки об'єкту. Наприклад, можете встановити **BevelInner = bvLowered** та **BevelOuter = bvRaised**. Подивіться, як зміниться вигляд панелі.

Після всіх попередніх налаштувань, залишилося написати код обробки кліку по кнопці. Як і в попередньому комп'ютерному практикумі, зробіть подвійний клік по кнопці на формі. Єдина інструкція, що дозволить, наприклад, перемножити введені числа буде мати вигляд:

```
Panel1->Caption = LabeledEdit1->Text + " * " +  
LabeledEdit2->Text + " = " +  
FloatToStr (StrToFloat(LabeledEdit1->Text) *  
StrToFloat(LabeledEdit2->Text));
```

На перший погляд, приведений код може здатися дуже складним. Спробуємо проаналізувати цей оператор. Початок повинен вже бути знайомим: властивості **Caption** компонента **Panel1** присвоюється значення виразу, який вказаний у правій частині від оператора «->».

Уся права частина виразу повинна бути типом рядка тексту. Починається рядок з тексту, який ввів користувач у вікно редагування **LabeledEdit1**. Уведений текст зберігається у властивості **Text** цього об'єкту. Тому, операцію **LabeledEdit1->Text** можна прочитати наступним чином: «отримати введений текст з об'єкту **LabeledEdit1**».

Далі використовується оператор «+». Для текстових типів даних оператор «+» виконує дію конкатенації рядків. Тобто, об'єднання (або склеювання) двох текстів в один. Так, наприклад, якщо написати операцію:

«Ілон » + «Маск»

то результатом даного запису буде – «Ілон Маск».

Тому, коли до тексту, взятого з **LabeledEdit1**, додається символ «*», це означає конкатенацію тексту з символом, тобто приєднання символу до тексту. Далі аналогічним образом додається текст з другого вікна редагування **LabeledEdit2** і символу «=».

Після цього потрібно вставити рядок результату множення двох цілих чисел. Цей результат буде числом. Тому, щоб додати його до вже наявного тексту, спочатку потрібно перетворити число в рядок. Дану операцію виконує функція **FloatToStr(...)**, яка перетворює заданий їй параметр в якості дійсного числа в рядок символів. Тобто, ця функція робить перетворення числа в текст.

Залишилася остання частина – отримати множення двох чисел. Однак, значення, що вводяться користувачем у вікно вводу об'єктів **LabeledEdit**, у властивості **Text**, зберігаються в вигляді текстів (наборів символів). Тому, перш ніж перемножувати, рядки символів потрібно перевести у числові дані. Ця операція виконується за допомогою функції **StrToFloat()**, яка перетворює символний рядок в значення типу дійсного числа. Знак «*», який вказаний між двома функціям **StrToFloat()**, означає операцію множення. Після компіляції, результат роботи програми буде виглядати як показано на рис. 2.3:

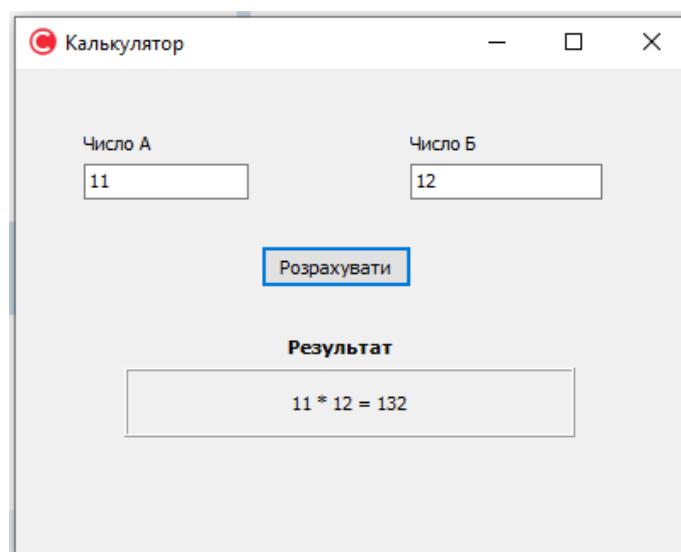


Рис. 2.3 – Результат виконання додатку множення двох чисел

Треба зазначити, що даний код можна записати у декілька простіших для розуміння інструкцій:

```
float a,b,sum;  
a=StrToFloat(LabeledEdit1->Text);  
b=StrToFloat(LabeledEdit2->Text);  
sum=a*b;  
Panel1->Caption= LabeledEdit1->Text + " * " +  
LabeledEdit2->Text + " = " +FloatToStr(sum);
```

Недоліком такого коду є використання трьох змінних a,b, sum, а перевагою – відносна простота для розуміння.

2.4. Завдання 3

Для непарних варіантів: модифікувати програму для розрахунку складного виразу із операторами * та -. Наприклад: $(A-B)*C$.

Для парних варіантів: модифікувати програму для розрахунку складного виразу із операторами + та /. Наприклад: $C/(A+B)$.

Для всіх варіантів: пропонується реалізувати програму, яка одночасно буде виконувати розрахунки для різних дій, як то: +,-,*,/.

Зазначені модифікації проробити як для консольних так і для графічних (у формі) додатків.

Приклад реалізації інтерфейсу у формі та консолі показаний на рис. 2.4.

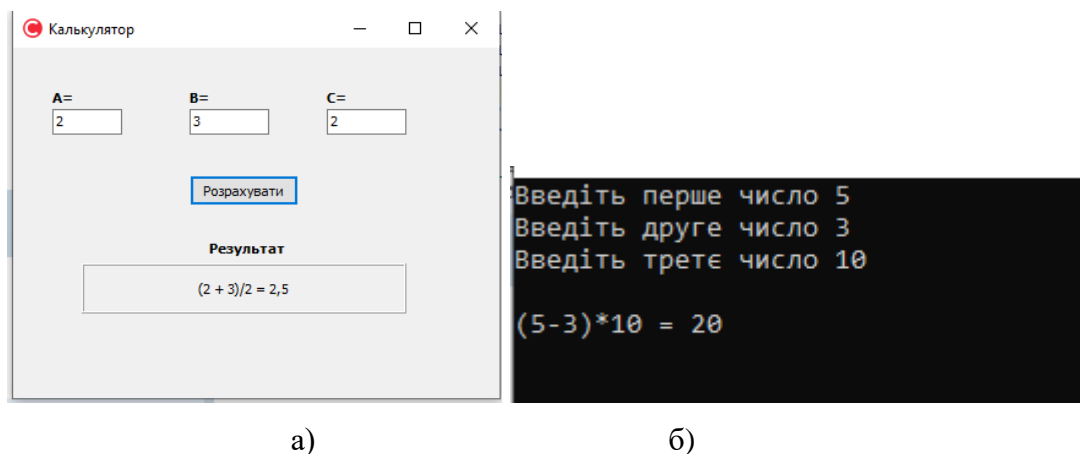


Рис. 2.4 – Приклади виконання завдання і інтерфейсу: а) у формі; б) в консолі

2.5. Зміст звіту

1. Код створених програм (перенести за допомогою Print Screen або скопіювавши). До коду бажано додавати коментарі, що пояснюють основні моменти і особливості виконання програми.
2. Фото із результатами виконання як в консолі, так і у формі.
3. Загальні висновки по роботі.

2.6. Контрольні запитання

1. Навіщо потрібна бібліотека *iostream*?
2. Який оператор використовується для отримання інформації із вхідного потоку?
3. Який оператор використовується для отримання інформації із вихідного потоку?
4. Які ви знаєте типи даних?
5. Що таке змінна?
6. Скільки разів можна використовувати оператор отримання з вихідного потоку в одному рядку коду?
7. Що таке приведення типу даних в C++? Наведіть приклади, як можна привести типи даних.
8. В яких випадках потрібно використовувати приведення типу даних?
9. Що таке конкатенація тексту? Який оператор використовується для цього?
10. Що таке **endl** та навіщо його використовувати?
11. Що таке керуючі символні послідовності? Наведіть приклади.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 3

ПРОГРАМУВАННЯ ЛІНІЙНИХ АЛГОРИТМІВ

Мета роботи: отримати навички створення додатків, які реалізовані за лінійними алгоритмами, для використання у різних сферах.

3.1. Теоретичні відомості

Для розрахунку математичних виразів на мові C/C++, є набір функцій, які зберігаються в стандартній бібліотеці математичних виразів. Для їх підключення потрібно оголосити заголовочний файл `math` (`cmath`), або `math.h`, в залежності від середовища програмування, яке використовується.

В табл. 3.1 наведено основні математичні(тригонометричні) оператори, які підключаються зі стандартної бібліотеки `math.h`:

Таблиця 3.1 – Функції для обчислення основних тригонометричних дій

№	Загальноприйняте позначення	Позначення на мові C++	Пояснення
1	$\sin x$	$\sin(x)$	синус числа x , в радіанах
2	$\cos x$	$\cos(x)$	косинус числа x , в радіанах
3	$tg x$	$\tan(x)$	тангенс числа x , в радіанах
4	$\arcsin x$	$\text{asin}(x)$	арксинус числа x , в радіанах
5	$\arccos x$	$\text{acos}(x)$	арккосинус числа x , в радіанах
6	$\text{arctg } x$	$\text{atan}(x)$	арктангенс числа x , в радіанах
7	π $\pi / 2$	M_PI M_PI_2	<u>константи:</u> значення чисел $\pi \approx 3.1416$, $\pi / 2 \approx 1.5708$
8	$ x $	$\text{fabs}(x)$, $\text{abs}(x)$	абсолютне значення x
9	\sqrt{x}	$\text{sqrt}(x)$	квадратний корінь із x
10	$\lg x$	$\log_{10}(x)$	десятковий логарифм числа x
11	$\ln x$	$\log(x)$	натуральний логарифм x
12	e^x	$\text{exp}(x)$	експонента x
13	x^y	$\text{pow}(x, y)$	x в ступені y

Як видно, бібліотека не описує деякі поширені функції, наприклад, логарифм за довільною основою, тому, для обрахунку цих значень необхідно користатися формулами перетворення. Тому в табл. 3.2 наведено деякі математичні формули, які можуть знадобитися під час виконання завдання:

Таблиця 3.2 – Формули перетворення

№	Формула
1	$ctg(x) = 1 / tg(x)$
2	$arcctg(x) = \pi / 2 - arctg(x)$
3	$\sqrt[n]{x} = x^{1/n}$
4	$\log_a(x) = \ln(x) / \ln(a)$
5	$1rad \times \frac{180}{\pi} = 57.296^\circ$

3.2. Завдання 1. Конвертор температур

Створіть програму переведення температур із градусів по Фаренгейту в градуси по Цельсію.

Порядок виконання

Спершу, створіть новий проект із графічним інтерфейсом, для цього виконайте [File->New->Windows VCL Application - C++ Builder](#).

Розмістіть на формі компоненти **Label1**, **Edit**, **Label2**, **Button1** і **Button2** із вкладки **Standard** палітри компонентів.

Об'єкт **Label** має властивість **WordWrap** – допустимість переносу слів довгого напису, яка перевищує довжину компонента, на новий рядок. Щоб такий перенос можна було виконати, потрібно встановити властивість **WordWrap** у стан **true**. Крім того, для властивості **AutoSize** встановити стан **false**, щоб розмір компонента не визначався розміром напису на ньому. Зробити висоту компонента такою, щоб мати можливість помістити декілька рядків.

Увага! Якщо **WordWrap** не визначено в стан **true** при **AutoSize** рівному **false**, то довгий текст, який не поміститься в область об'єкту, буде обрізатися у відповідності до розміру компонента.

Далі потрібно перейти в обробник події **OnClick**, зробивши подвійний клік на компоненті **Button1** на формі.

При введенні із вікна числової інформації потрібно використовувати наступні функції:

- **StrToInt(s)** – функція перетворення рядка **s** в числове значення цілочисельного типу;
- **StrToFloat(s)** – функція перетворення рядка **s** в числове значення з плаваючою комою, тобто дійсне число.

Виконайте перетворення температур за формулою:

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

де **C** – це температура за шкалою Цельсія, **F** – температура за шкалою Фаренгейта. Попередньо, необхідно оголосити всі відповідні змінні.

Після розрахунків потрібно вивести в об'єкт **Label2** змішану інформацію, яка складається з рядків символів та чисел. У виводі результату кількість символів після коми потрібно обмежити 4.

У попередніх роботах використовувалася функція **FloatToStr(s)**, щоб привести типи даних з числового типу в символьний. Однак, якщо застосувати цю функцію для розрахунку дійсних чисел, кількість знаків після коми може бути надто великою. Тому, щоб мати можливість переводити дані у більш зручний вигляд, використовується модифікована функція перетворення **FloatToStrF(...)** функція *форматованого* виводу. Виклик цієї функції виконується у вигляді **FloatToStrF(a, b, c, d)**, де:

a – це число з плаваючою комою (дійсне число), яке необхідно привести до символьного типу у відповідності до певного формату;

b – це формат, який визначає спосіб зображення (вигляд) числа.

Можливі варіанти:

- **ffGeneral** – універсальний;

- **ffExponent** – науковий;
- **ffFixed** – з фіксованою комою;
- **ffNumber** – з роздільниками груп розрядів;
- **ffCurrency** – фінансовий.

c – це точність, що вказує на необхідну загальну кількість цифр;

d – це кількість цифр після десяткової коми.

Тобто, для того, щоб обмежитись виведенням двох символів після коми числа, що збережено у змінній *Z*, необхідно викликати функцію `FloatToStrF` з наступними параметрами:

```
FloatToStrF(Z, ffFixed, 6, 2);
```

Нагадаємо, що для формування тексту, що складається із декількох фрагментів, можна використовувати оператор «+», який для рядків тексту означає їх склеювання (конкатенацію, об'єднання).

Загальний вигляд готової програми має бути як на рис. 3.1.

Як можна побачити, при взаємодії із додатком у поле компонента **Edit** можна вводити не лише числа, а й різні символи, що при виконанні програми стане причиною критичних помилок. Тому пропонується модифікувати створений додаток, щоб обмежити введення літер і дозволити введення лише чисел.

Для цього потрібно помістити наступний код в поле блоку коду **OnKeyPress** відповідного **Edit**:

```
// Key – код натиснутої клавіші
// перевірка, чи є допустимим символ
    if ((Key >= '0') && (Key <= '9')) // цифра
        return;
    else if ((Key == '.') || (Key == ','))
        {
            // DecimalSeparator – глобальна змінна – роздільник цілої та дробової
частин (десяткова кома), містить символ, який використовується в якості
роздільника при запису дробових чисел.
if ((Edit1->Text).Pos(FormatSettings.DecimalSeparator) !=
0)

                Key = 0; // роздільник вже введений
            else // якщо ще ні
                Key = FormatSettings.DecimalSeparator;
```



```

        return;
    }
    if (Key == VK_BACK) // клавіша <Backspace>
        return;
    if (Key == VK_RETURN) // клавіша <Enter>
    {
        Edit1->SetFocus(); // встановлюємо курсор сюди
        return;
    }
    // інші клавіші заборонені
    Key = 0; // код заборонених символів замінено нулем, в результаті
    символ в поле редагування не буде відображений

```

Код захисту від введення літер приведений «як є», тому не є обов'язковим для виконання.

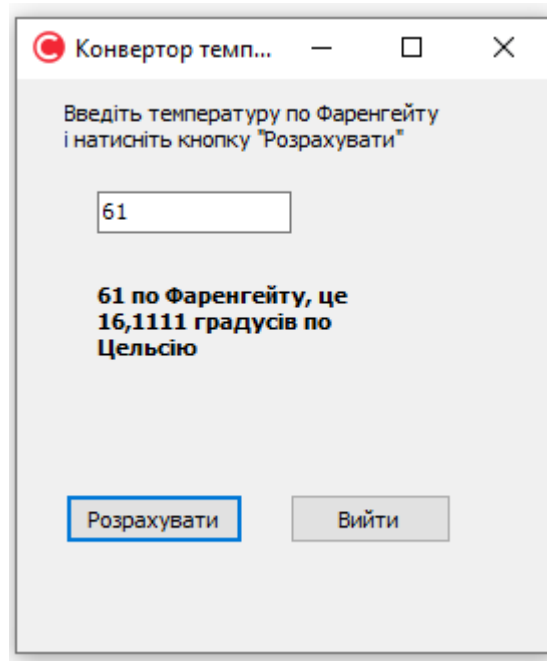


Рис. 3.1 – Загальний вигляд програми «Конвертор»

Для того, щоб після натискання на кнопку «Вийти», форма закрилась, потрібно використати метод **Close()**, який закриває форму.

Тепер, виконаємо це ж завдання в консольному додатку. Приклад реалізації інтерфейсу у консольному додатку показаний на рис. 3.2.

```

Введіть температуру по Фаренгейту 51
51 градусів по Фаренгейту = 10.5556
градусів по Цельсію

```

Рис. 3.2 – Вигляд інтерфейсу програми «Конвертор» у консолі

3.3. Завдання 2. Обрахунок за математичними формулами

Потрібно створити програму розрахунку за двома формулами. Результат розрахунку по першій формулі повинен співпадати з результатом другої формули. *Зауваження.* Для деяких варіантів таке співвідношення між формулами може не виконуватися, у такому випадку пропонується перевірити правильність розрахунків використовуючи калькулятор.

Якщо номер по списку більше 20, то починати рахувати спочатку, тобто 21-й варіант = 1, 22 = 2 і т. д.

Варіанти завдань:

Варіант 1

$$z_1 = 2 \sin^2(3\pi - 2\alpha) \cos^2(5\pi + 2\alpha)$$

$$z_2 = \frac{1}{4} - \frac{1}{4} \sin\left(\frac{5}{2}\pi - 8\alpha\right)$$

Варіант 2

$$z_1 = \cos \alpha + \sin \alpha + \cos 3\alpha + \sin 3\alpha$$

$$z_2 = 2\sqrt{2} \cos \alpha \cdot \sin\left(\frac{\pi}{4} + 2\alpha\right)$$

Варіант 3

$$z_1 = \frac{(\sin 2\alpha + \sin 5\alpha - \sin 3\alpha)}{(\cos \alpha + 1 - 2 \sin^2 \alpha)}$$

$$z_2 = 2 \sin \alpha$$

Варіант 6

$$z_1 = \cos \alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha$$

$$z_2 = 4 \cos \frac{\alpha}{2} \cdot \cos \frac{5\alpha}{2} \cdot \cos 4\alpha$$

Варіант 7

$$z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right)$$

$$z_2 = \frac{\sqrt{2}}{2} \sin \frac{\alpha}{2}$$

Варіант 8

$$z_1 = \cos^4 x + \sin^2 y + \frac{1}{4} \sin^2 2x - 1$$

Варіант 4

$$z_1 = \frac{(m-1)\sqrt{m} - (n-1)\sqrt{n}}{\sqrt{m^3n + nm + m^2 - m}}$$

$$z_2 = \frac{\sqrt{m} - \sqrt{n}}{m}$$

Варіант 5

$$z_1 = 1 - \frac{1}{4} \sin^2 2\alpha + \cos 2\alpha$$

$$z_2 = \cos^2 \alpha + \cos^4 \alpha$$

$$z_2 = \sin(x+y) \sin(x-y)$$

Варіант 9

$$z_1 = (\cos \alpha - \cos \beta)^2 - (\sin \alpha - \sin \beta)^2$$

$$z_2 = -4 \sin^2 \frac{\alpha - \beta}{2} \cos(\alpha + \beta)$$

Варіант 10

$$z_1 = \frac{\sin\left(\frac{\pi}{2} + 3d\right)}{1 - \sin(3\alpha - \pi)}$$

$$z_2 = \operatorname{ctg}\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right)$$

Варіант 11

$$z_1 = \frac{1 - 2 \sin^2 \alpha}{1 + \sin 2\alpha}$$

$$z_2 = \frac{1 - \operatorname{tg} \alpha}{1 + \operatorname{tg} \alpha}$$

Варіант 16

$$z_1 = \frac{x^2 + 2x - 3 + (x+1)\sqrt{x^2 - 9}}{x^2 - 2x - 3 + (x-1)\sqrt{x^2 - 9}}$$

$$z_2 = \sqrt{\frac{x+3}{x-3}}$$

Варіант 12

$$z_1 = \frac{\sin 4\alpha}{1 + \cos 4\alpha} \cdot \frac{\cos 2\alpha}{1 + \cos 2\alpha}$$

$$z_2 = \operatorname{ctg}\left(\frac{3}{2}\pi - \alpha\right)$$

Варіант 17

$$z_1 = \frac{\sqrt{(3m+2)^2 - 24m}}{3\sqrt{m} - \frac{2}{\sqrt{m}}}$$

$$z_2 = -\sqrt{m}$$

Варіант 13

$$z_1 = \frac{\sin \alpha + \cos(2\beta - \alpha)}{\cos \alpha - \sin(2\beta - \alpha)}$$

Варіант 18

$$z_1 = \left(\frac{a+2}{\sqrt{2a}} - \frac{a}{\sqrt{2a+2}} + \frac{2}{2-\sqrt{2a}}\right) \frac{\sqrt{a}-\sqrt{2}}{a+2}$$

$$z_2 = \frac{1 + \sin 2\beta}{\cos 2\beta}$$

Варіант 14

$$z_1 = \frac{\cos \alpha + \sin \alpha}{\cos \alpha - \sin \alpha}$$

$$z_2 = \operatorname{tg} 2\alpha + \operatorname{sec} 2\alpha$$

Варіант 15

$$z_1 = \frac{\sqrt{2b + 2\sqrt{b^2 - 4}}}{\sqrt{b^2 - 4} + b + 2}$$

$$z_2 = \frac{1}{\sqrt{b + 2}}$$

$$z_2 = \frac{1}{\sqrt{a} + \sqrt{2}}$$

Варіант 19

$$z_1 = \left(\frac{1 + a + a^2}{2a + a^2} + 2 - \frac{1 - a + a^2}{2a - a^2} \right)^{-1} (5 - 2a^2)$$

$$z_2 = \frac{4 - a^2}{2}$$

Варіант 20

$$z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos \alpha - \cos 3\alpha + \cos 5\alpha}$$

$$z_2 = \operatorname{tg} 3\alpha$$

Результати виконання та загальний вигляд інтерфейсу програми показані на рис. 3.3.

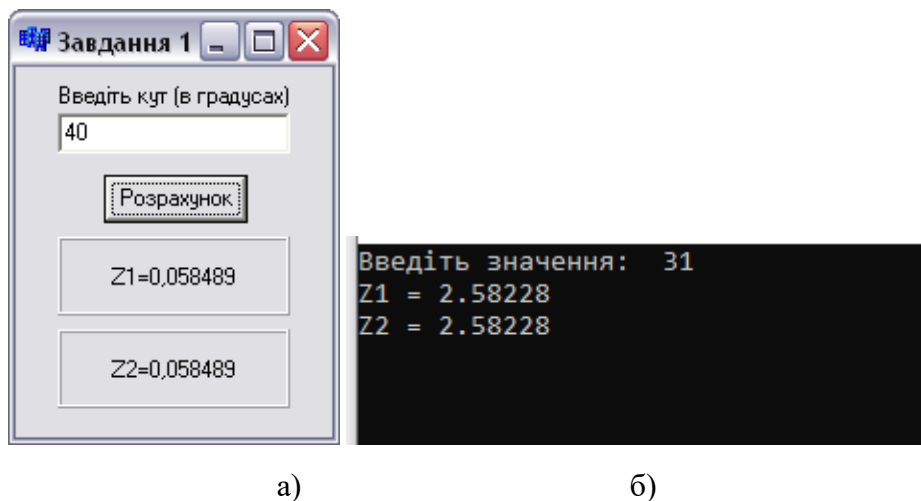


Рис. 3.3 – Результат виконання завдання: а) у формі; б) в консолі

3.4. Завдання 3. Обрахунок за умовою задачі

Створіть програму для вирішення зазначених задач. Номер задачі відповідає номеру варіанту. Якщо номер по списку більше 20, то починати рахувати спочатку, тобто 21-й = 1 задача і т. д. Інтерфейс розробляється на власний розсуд, проте він повинен бути функціональним та зрозумілим.

Програми треба реалізовувати як в консолі, так і із графічним інтерфейсом, тобто у формі.

Варіанти завдань:

1. Дано круглу квіткову клумбу. Обчислити її периметр і площу за заданим радіусом.
2. Обчислити площу і периметр прямокутного трикутника, якщо задані значення катету та гострий кут.
3. Обчислити довжину та площу кола за заданим радіусом.
4. Ділянка поля має форму рівнобічної трапеції. Обчислити її периметр і площу за заданими сторонами.
5. Ресторан заковує щодня масло m_1 (кг) по 40,90 грн за кілограм, майонез m_2 (кг) по 62,40 грн, вершки m_3 (кг) по 144,10 грн. Визначити суми, потрібні для купівлі окремих продуктів, і загальну суму.
6. Скільки секунд в N годинах, добах, роках?
7. Обчислити кінетичну $E = mv^2/2$ та потенціальну $p = mgh$ енергії тіла заданої маси m , яке рухається на висоті h зі швидкістю v .
8. Ціни на цукор, гречку та олію зросли на p відсотків. Вивести старі та нові ціни.
9. Обчислити площу поверхні $S = 4\pi r^2$ та об'єм $V = 4\pi r^3/3$ сфери за заданим радіусом r .
10. Є три сорти кукурудзи із різною врожайністю (наприклад, 44 т/га, 21 т/га, 60 т/га) і площі трьох відповідних полів (га). Скільки зібрали кукурудзи з кожного поля і сумарно?
11. Швидкість світла 299792 км/с. Яку відстань долає світло за N годин та N діб?
12. Обчислити площу поверхні місяця $S = 4\pi r^2$ та об'єм супутника $V = 4\pi r^3/3$, якщо відомо, що його радіус 1740 км.
13. Обчислити гіпотенузу та площу прямокутного трикутника за заданими двома катетами.

14. Обчислити об'єм куба та площу його бічної поверхні, якщо відома довжина ребра.
15. Увести продуктивності роботи трьох труб М, N, К (л/хв), які наповнюють басейн, і час їхньої роботи (хв), окремо для кожної труби. Скільки води набрано в басейні?
16. Квадрат вписаний в коло із радіусом r , знайти площу і периметр квадрата.
17. Визначити пройдений тілом шлях $h = gt^2/2$, що падає із прискоренням g , після першої та N ої секунди падіння.
18. Телефонні розмови по трьом різних тарифах коштують а, b, c (грн/хв). Тривалість розмов t_a, t_b, t_c хв., відповідно. Яку суму нарахує оператор по кожному тарифу. Скільки всього грошей отримає оператор?
19. Вивести значення кутів прямокутного трикутника та його площу, якщо відомі значення його катетів.
20. Дана циліндрична ємність висотою h та радіусом основи r . Обчислити площу бічної поверхні $S = 2\pi rh$ та її об'єм $V = \pi r^2 h$. Результати виконання та загальний вигляд інтерфейсу програми показані на рис. 3.4.

Введіть радіус

1

Розрахунок

Периметр клумби: 6,28 м

Площа клумби: 3,14 м²

Рис. 3.4 – Результат обчислення задачі у формі

3.5. Зміст звіту

1. Код створених програм (перенести за допомогою Print Screen або скопіювавши). До коду бажано додавати коментарі, що пояснюють основні моменти і особливості виконання програми.
2. Фото із результатами виконання як в консолі, так і у формі.
3. Загальні висновки по роботі.

3.6. Контрольні запитання

1. Яка бібліотека слугує для підключення математичних виразів?
2. Чим відрізняється $\text{fabs}(x)$ від $\text{abs}(x)$?
3. Як в C++ записати вираз для експоненти?
4. Як позначається константа π в C++?
6. Наведіть приклади оголошення змінної. У чому принципова різниця?
7. Який вираз використовується в C++ для піднесення до степеня?
8. Які є способи, щоб зробити затримку виконання програми?
9. Чим відрізняються типи даних *float* від *double*?
10. Що таке приведення типів даних?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 4

СТВОРЕННЯ ДОДАТКА З ДЕКІЛЬКОМА ФОРМАМИ

Мета роботи: отримати навички створення додатків із декількома формами, оглянути основні принципи передачі інформації між формами.

4.1. Теоретичні відомості

Оскільки для виконання завдання даного комп'ютерного практикуму не будуть використані принципово нові функції та конструкції, то основні особливості створення додатку із декількома формами у проекті, їх взаємодії і т. п. будуть детально розглянуті безпосередньо під час виконання завдання.

Отже, для прикладу створимо програму з графічним інтерфейсом і декількома формами.

Для цього будемо виконувати це поетапно, у відповідності до зазначеного алгоритму дій:

1. Спершу, створіть новий проект із графічним інтерфейсом, для цього виконайте **File-> New->Windows VCL Application - C++ Builder**.

2. Змініть значення властивості **Name** на **MainForm**, а властивість **Caption** – на **Multiple Forms Test Program**. Одночасно, проаналізуйте у чому принципові відмінності цих властивостей.

3. Збережіть проект. Назвіть модуль ім'ям **Main**, а проект – ім'ям **Multiple**.

4. Тепер розмістіть на формі кнопку. Надайте властивості **Name** значення **ShowForm2**, а властивості **Caption** – значення **Показати другу форму**.

5. Оберіть в головному меню пункт **File-> New-> VCL Form - C++ Builder**, щоб створити нову форму. Після створення, нова форма буде мати ім'я **FormX** і розміститься поверх головної форми. Потрібно, щоб нова форма мала менший розмір і не перекривала видимість основної форми. Змініть розмір і положення нової форми так, щоб вона була приблизно в 2 рази менше головної форми та розташовувалася в її центрі. Для переміщення

форми використовуйте рядок заголовку. Розмір можна змінити за рахунок перетягування нижнього правого кута.

6. Змініть значення властивості **Name** нової форми на **SecondForm**, а її властивість **Caption** – на **A Second Form**.

7. Оберіть в головному меню пункт **File -> Save** (або натисніть на кнопку **Save** на панелі інструментів) та збережіть файл під назвою **Second**.

8. Оберіть компонент **Label** і розташуйте його в новій формі. Змініть текст властивості **Caption** на **Це друга форма**. Змініть розмір і колір тексту. Відцентруйте повідомлення відносно форми, загальний вигляд проекту має бути як показано на рис. 4.1.

9. Оберіть головну форму. Зверніть увагу, що друга форма тепер закрита головною формою. Зробіть подвійний клік по кнопці **Show Form 2**. На екрані з'явиться вікно редактора коду і курсор буде розміщений як раз там, де потрібно вводити текст.

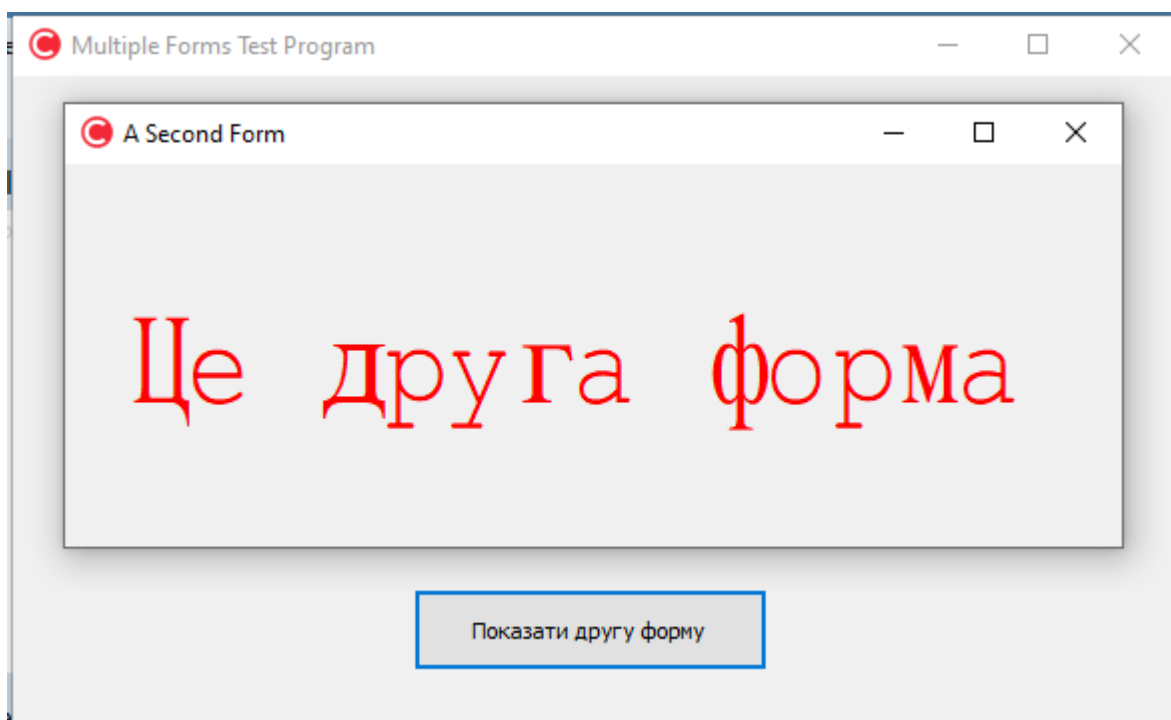


Рис. 4.1 – Загальний вигляд додатку із декількома формами

10. Введіть команду, яка приведена нижче (вам потрібно написати лише один рядок, що стоїть у блоці коду між фігурними дужками):

```

- //-----
- void __fastcall TMainForm::Button1Click(TObject *Sender)
- {
20 |   SecondForm->ShowModal();
- | }
- //-----

```

11. Запустіть програму.

Після запуску було отримано повідомлення про похибку **Undefined symbol 'SecondForm'** (Невизначений символ 'SecondForm').

Давайте згадаємо, що у нас є 2 вихідних файли з відповідними заголовками. Проте, в модулі **MainForm** немає оголошення змінної **SecondForm**, яка насправді є вказівником екземпляру класу **TSecondForm**. Необхідно вказати, де розміщено оголошення цього класу. Для цього потрібно включити заголовочний файл для **SecondForm** в початковий файл **MainForm** за допомогою директиви **#include**. Перейдіть у вікно редактору коду та клацніть на вкладку **Main.cpp** для відображення модуля головної форми. Перейдіть до початку файлу. Перші декілька рядків повинні мати наступний вигляд:

```

- //-----
- #include <vcl.h>
- #pragma hdrstop
- #include "Main.h"
- //-----

```

Можна побачити, що сюди включено заголовочний файл **Main.h**, але немає файлу **Second.h**. Для його підключення зробимо наступне:

12. Оберіть в головному меню пункт **File-> Use Unit**. На екрані з'явиться діалогове вікно **Use Unit**, яке показано на рис. 4.2. Клацніть на ім'я **Second**, а потім на кнопку ОК, щоб закрити діалогове вікно.

Діалогове вікно **UseUnit** відображає лише ті модулі проекту, які ще до нього не включені. Додані модулі не містяться в списку. Після кліку по кнопці ОК середовище розробки автоматично додає у файл директиву **#include** для **Second.h**:

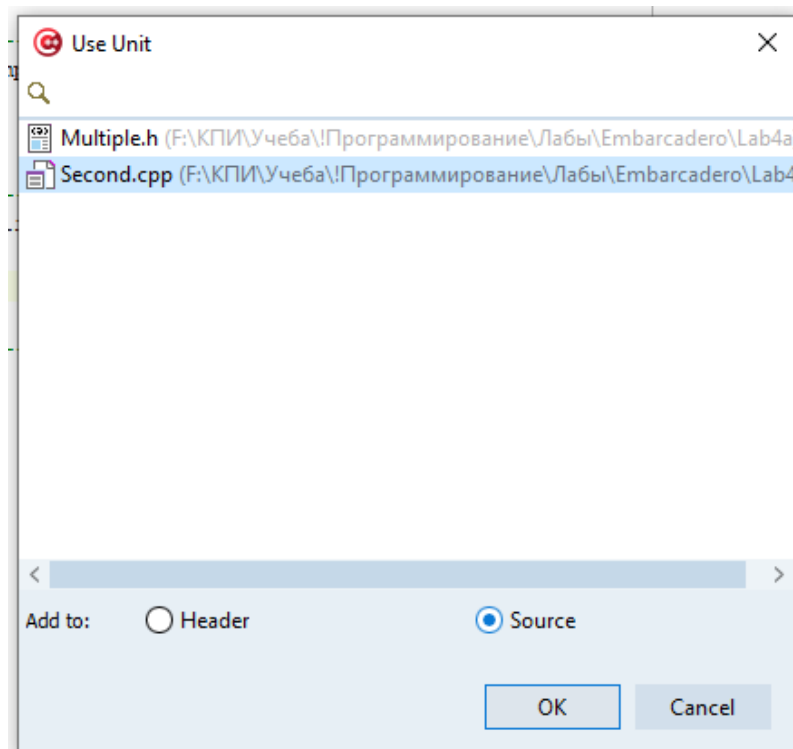


Рис. 4.2 – Діалогове вікно **Use Unit**

```

#include <vcl.h>
#pragma hdrstop

#include "Main.h"
#include "Second.h"

```

13. Запустіть програму. На цей раз компіляція пройде без помилок і програма запрацює. Клік по кнопці [Показати другу форму](#) призведе до появи другої форми на екрані.

Тепер розберемось як передавати значення із однієї форми у іншу. Для цього, на першу та другу форми додамо два об'єкта типу **Edit**. І модифікуємо код наступним чином:

```

//-----
void __fastcall TMainForm::Button1Click(TObject *Sender)
{
    SecondForm->Edit1->Text= MainForm->Edit1->Text;
    SecondForm->ShowModal();
}
//-----

```

Оскільки цей код пишеться у файлі Main.cpp, що стосується головної форми, то код можна спростити, прибравши [MainForm](#):

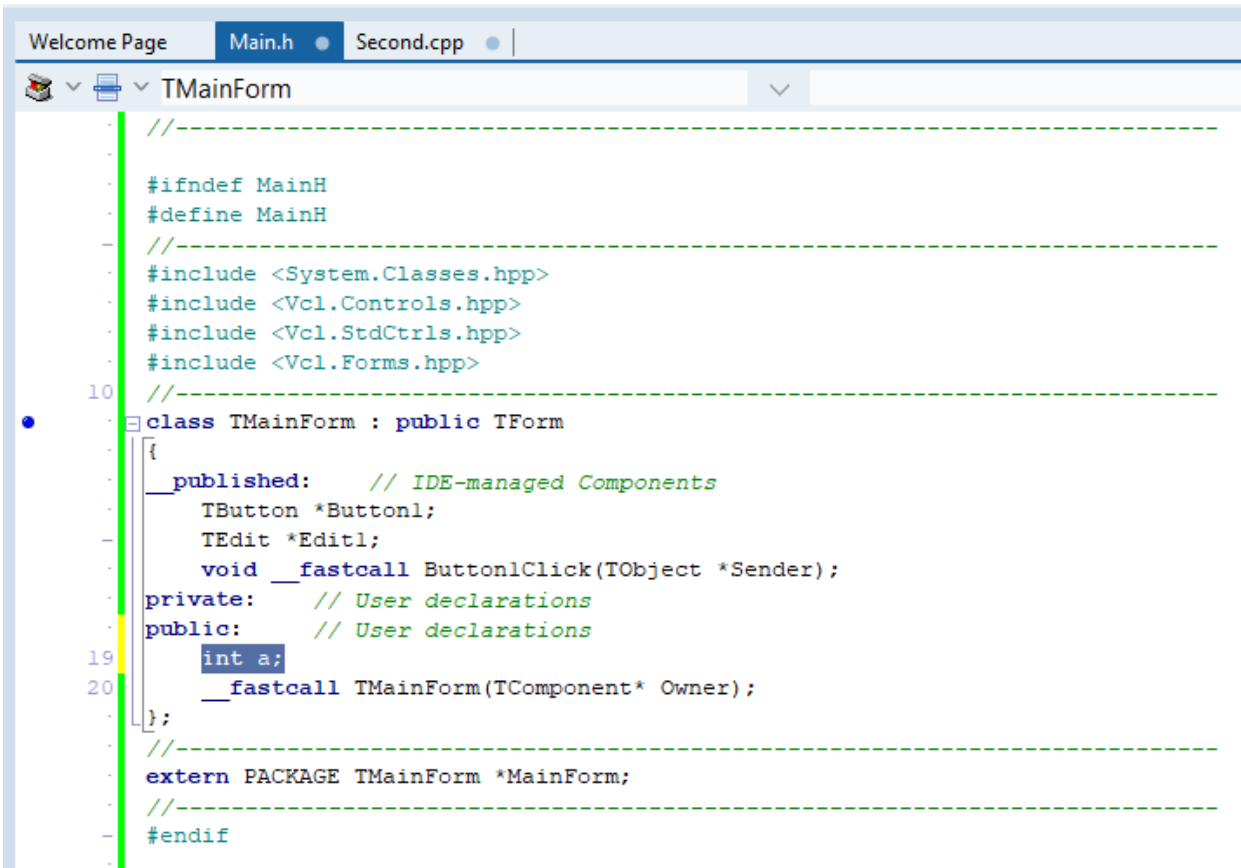
```
SecondForm->Edit1->Text= Edit1->Text;
```

Причому, не дивлячись на однакові назви об'єктів *Edit1* – вони різні, оскільки належать різним формам.

Якщо потрібно реалізувати двосторонню передачу даних, тобто із першої форми у другу і навпаки, то для цього треба підключити файл [Main.h](#) директивою `#include " Main.h "` у файлі [Second.cpp](#), як показано вище.

Також рекомендовано змінити метод [ShowModal\(\)](#); на [Show\(\)](#);, це дозволить вільно перемикались між формами не закриваючи активну.

Доволі часто бувають випадки, що значення із однієї форми у іншу передаються через буферні змінні, такі змінні потрібно оголошувати у заголовочному файлі [Main.h](#) у розділі `public`: як показано на рис. 4.3.



```
//-----  
#ifndef MainH  
#define MainH  
//-----  
#include <System.Classes.hpp>  
#include <Vcl.Controls.hpp>  
#include <Vcl.StdCtrls.hpp>  
#include <Vcl.Forms.hpp>  
10 //-----  
class TMainForm : public TForm  
{  
  __published: // IDE-managed Components  
    TButton *Button1;  
    TEdit *Edit1;  
    void __fastcall Button1Click(TObject *Sender);  
private: // User declarations  
public: // User declarations  
19   int a;  
20   __fastcall TMainForm(TComponent* Owner);  
};  
//-----  
extern PACKAGE TMainForm *MainForm;  
//-----  
#endif
```

Рис. 4.3 – Оголошення змінної у Main.h

4.2. Завдання

Створити додаток із декількома формами і організувати послідовність дій у відповідності до завдання:

Варіанти завдань:

1. Є дві форми, організувати двосторонню передачу інформації у поля типу *Edit*.
2. Є три форми, організувати передачу даних із першої форми у другу, а із другої – у третю.
3. Є три форми, дані, що введені у другу і в третю форму по натисканню на кнопку передаються у першу.
4. Є дві форми, організувати двосторонню передачу інформації у поля типу *Edit*, використовуючи буферну змінну.
5. Є три форми, дані, що введені у поле *Edit* першої форми, з'являються у другій та третій формах.
6. Є три форми, дані, що введені у другу і в третю форму по натисканню на кнопку передаються у першу, використовувати буферні змінні.
7. Є дві форми, при введенні значення у поле *Edit* першої форми і натисканні на кнопку, у другій формі з'являється це значення +10.
8. Є три форми, дані, що введені у першу і третю форми - передаються у другу.
9. Є три форми, число, що введено у поле *Edit* першої форми, з'являються у другій формі збільшене на 1, а у третій – на два.
10. Є чотири форми, дані, що введені у першій формі, з'являються у всіх інших формах одночасно.

Якщо номер за списком більше 10, то починати рахувати спочатку, тобто 11й = 1 задача і т. д. Інтерфейс розробляється на власний розсуд, проте він повинен бути функціональним та відповідати завданню.

4.3. Зміст звіту

1. Код створених програм (перенести за допомогою Print Screen або скопіювавши). До коду бажано додавати коментарі, що пояснюють основні моменти і особливості виконання програми.
2. Фото із результатами виконання.
3. Загальні висновки по роботі.

4.4. Контрольні запитання

1. У яких випадках зручно використовувати декілька форм? Наведіть приклади.
2. Як додати до проекту декілька форм?
3. Навіщо використовується меню File-> Use Unit?
4. Як передати данні із одного об'єкту однієї форми у інший об'єкт другої форми?
5. Де потрібно оголошувати змінні, щоб вони були доступні для декількох форм? Як до них звертатись?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 5

ЗНАЙОМСТВО З ІНСТРУКЦІЄЮ ВИБОРУ

Мета роботи: отримати навички роботи із інструкцією вибору if.

5.1. Теоретичні відомості

Інструкція вибору, або умовний оператор if використовується у відповідності до загального синтаксису:

```
if (умова) інструкція;  
    else інструкція;  
або так  
    if (умова)  
        {  
            послідовність інструкцій;  
        }  
    else  
        {  
            послідовність інструкцій;  
        }
```

Розберемо на прикладі коду, що визначає стан здоров'я пацієнта по його температурі:

```
float temp;  
cout<<"Яка у вас температура ?\n" ;  
cin>>temp;  
    if (temp<37)  
        cout<<" Ви здорові ";  
    else cout<<" Вам потрібно до лікаря " ;  
    getch();
```

Отже, рядок `if (temp<37)` означає, що якщо змінна `temp` прийме значення менше ніж 37 градусів, то виконається інструкція `cout<<" Ви здорові "`; у протилежному випадку виконається гілка `else` з інструкцією `cout<<" Вам потрібно до лікаря " ;`

Проте, можна зауважити, що температура нижча за, наприклад, 35 градусів, також показник певного захворювання, тому умову можна модифікувати так: `if (temp>=35 && temp<37)`. Це буде означати, що інструкція `cout<<" Ви здорові "`; буде виконана лише тоді коли виконуються одразу дві умови `temp>=35` і `temp<37`.

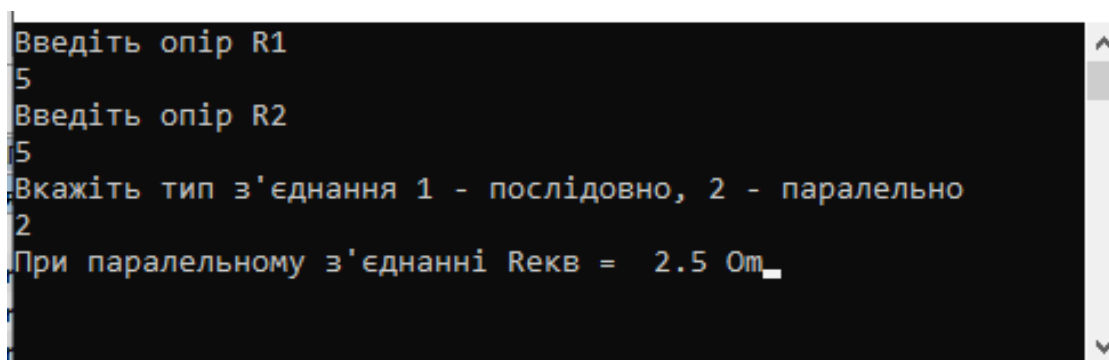
Такий приклад демонструє те, що інструкція `if` дозволяє реалізовувати доволі складні логічні умови із використанням операторів як порівняння (`>`, `<`, `==`, `!=` ...) так і логічних (`&&`, `||`, `!`).

5.2. Завдання 1. Розрахунок опорів

Створіть програму, що обчислює опір електричної ланки, яка складається з двох резисторів. Резистори можуть бути з'єднані послідовно, або паралельно.

Нагадаємо, що при послідовному з'єднанні опорів, їх еквівалентний опір розраховується як сума опорів, тобто $R_{\text{екв}} = R_1 + R_2 + \dots$. Для паралельного з'єднання еквівалентний опір буде обрахований за формулою $(1/R_{\text{екв}}) = 1/R_1 + 1/R_2 + \dots$.

Для консольного додатку інтерфейс реалізувати як показано на рис. 5.1.



```
Введіть опір R1
5
Введіть опір R2
5
Вкажіть тип з'єднання 1 - послідовно, 2 - паралельно
2
При паралельному з'єднанні R_екв = 2.5 Ом_
```

Рис. 5.1 – Загальний вигляд інтерфейсу консольного додатку для завдання 1.

Для реалізації додатку із графічним інтерфейсом запропоновано використати об'єкти **RadioButton** та **Radiogroup**. Загальний вигляд додатку показано на рис. 5.2.

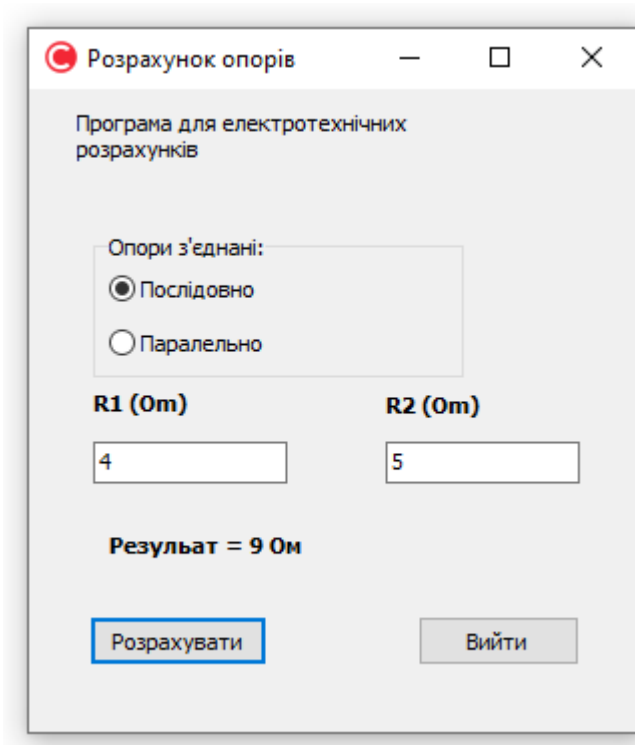


Рис. 5.2 – Загальний вигляд графічного інтерфейсу для завдання 1.

5.3. Завдання 2. Вирішення задачі за умовою

Створити програму для вирішення поставленої задачі за умовами. Номер задачі відповідає номеру варіанту. Якщо номер за списком більше 15, то починати рахувати спочатку, тобто 16й = 1 задача і т.д. Інтерфейс розробляється на власний розсуд, проте він повинен бути функціональним та зрозумілим.

Завдання реалізовувати і в консольному додатку, і в формі!

Після завдання, жирним шрифтом наведено рекомендований вид екрану під час роботи консольного додатку.

Варіанти завдань:

1. Написати програму, яка вмє правильно відмінювати числа від 1 до 999, що позначають грошову одиницю (гривні). Наприклад 1 гривня, 10 гривень, 921 гривня т. ін.

Скільки гривень це коштує? 921

921 гривня

2. Написати програму, яка вміє правильно відмінювати числа від 1 до 999, що позначають грошову одиницю (долари). Наприклад 1 долар, 2 долари, 100 доларів т. ін.

Скільки доларів ви хочете обміняти? 120

120 доларів

3. Написати програму, яка правильно відмінює число копійок, що введено з клавіатури. Діапазон введених значень має бути від 1 до 99. Наприклад: 5 копійок, 41 копійка і т.п.

Скільки копійок? 21

21 копійка

4. Написати програму, яка визначає робочий день чи вихідний, за введеним з клавіатури номером: "Робочий день", "Субота" або "Неділя".

Який сьогодні день? (введіть номер дня тижня) 4

Робочий день.

5. Напишіть програму, яка визначає пору року, в залежності від уведеного номера місяця.

Введіть номер місяця (число від 1 до 12): 12

Зима. Брррр!!!

6. Написати програму, яка обчислює оптимальну вагу для користувача, порівнює його з реальною і видає рекомендацію про необхідність набрати вагу або схуднути. Оптимальна вага обчислюється за формулою: для чоловіків - зріст (см) - 95; для жінок зріст (см) - 110.

Ви (Ч/Ж) Ж

Ваш зріст: 170

Ваша вага: 58

Вам треба поправитися на 2.00 кг.

7. Написати програму обчислення вартості розмови по телефону в різних регіонах. Регіон 1 – повна вартість, регіон 2 – знижка 20%, регіон 3 – знижка 50%. Обчислення вартості розмови по телефону.

Введіть вихідні дані:

Тривалість розмови (кількість хв.): 3

Вартість хвилини (грн.): 1,5

Введіть регіон обслуговування (1,2,3): 2

Вартість розмови: 3,6 грн (Надається знижка 20%.)

8. Написати програму, яка відповідає, чи ділиться на X введене з клавіатури ціле число без остачі.

Введіть ціле число: 778

На яке число ділимо?: 7

Число 778 без остачі на 7 не ділиться.

9. Написати програму, яка перевіряє, чи є введене користувачем ціле число парним і виводить сусідні числа.

Введіть ціле число: 23

Число 23 - непарне.

Сусідні числа 22 та 24.

10. Створити програму, що виводить на екран найближче до 0 та до 100 з двох чисел, записаних в змінні a і b .

Введіть число a : 12

Введіть число b : 60

Число 12 - ближче до 0.

Число 60 - ближче до 100.

11. Напишіть програму перевірки знань з історії. Програма повинна вивести питання і три варіанти відповіді. Користувач повинен вибрати правильну відповідь і ввести його номер.

Дніпропетровськ отримав свою назву від:

1. Річки Дніпро

2. На прізвище державного діяча Г. І. Петровського

3. 1 і 2 варіанти

Введіть номер правильної відповіді: 1

Ви помилилися.

Правильна відповідь: 3.

12. Напишіть програму, що переводить секунди у години і хвилини.
Введіть значення в секундах: 3676
1 год 1 хв.
13. Створити програму, що шукає найбільшу цифру у тризначному натуральному числі.
Введіть ваше тризначне число 123
Найбільша цифра 3
14. У три змінні a, b і c записуються три цілих попарно нерівних між собою числа. Створити програму, яка переставляє числа в змінних таким чином, щоб при виведенні на екран послідовність a, b і c виявилася зростаючою:
Числа в змінних a, b і c: 3 9 -1
Зростаюча послідовність: -1 3 9
15. Написати програму, яка обчислює дату наступного дня. Враховуючи високосні роки.
Введіть дату (число, місяць, рік) 31 12 2021
Завтра 1.1.2022
З наступаючим Новим роком!

5.4. Зміст звіту

1. Код створених програм (перенести за допомогою Print Screen або скопіювавши). До коду бажано додавати коментарі, що пояснюють основні моменти і особливості виконання програми.
2. Фото із результатами виконання як в консолі, так і у формі.
3. Загальні висновки по роботі.

5.5. Контрольні запитання

1. Опишіть загальний формат інструкції if.
2. Чи обов'язково завжди використовувати else? Обґрунтуйте із прикладом.
3. Чи вірно те, що умова, що керує if-інструкцією, повинна обов'язково використовувати оператор відношення?

4. У яких випадках можна використовувати тернарний оператор «?» замість інструкції if?
5. У якому випадку умова if(a=5) виконається? Чому?
6. Чи можна використовувати у умові if логічні оператори?
7. Яка принципова різниця між об'єктами RadioButton та Radiogroup? Коли і який об'єкт зручніше застосовувати?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 6

ПРОГРАМУВАННЯ РОЗГАЛУЖЕНИХ АЛГОРИТМІВ

Мета роботи: отримати навички роботи із інструкціями вибору типу if-else-if та switch.

6.1. Теоретичні відомості

Не дивлячись на те, що оператор вибору if/else використовується у 96% відсотках програм, часто не можна розділити умову лише на значення true або false. У такому випадку використовується так звана сходинкова конструкція типу if-else-if, або switch. Тепер детально розглянемо кожну з них. Загальний синтаксис інструкції if-else-if виглядає наступним чином:

```
if (умова)
{інструкції; }
else if (умова)
{інструкції; }
else if (умова)
{інструкції; }
*
*
else {інструкції; }
```

Така конструкція використовується коли є декілька взаємопов'язаних варіантів подій. Для прикладу розглянемо тестову програму, що визначає пору року по середньодобовій температурі:

```
cin>>t;
if (t>20)
cout<<"Тепло ";
else if (t<-10)
cout<<"Брррр...Зима ";
else cout<<"Або весна, або осінь 😊";
```

Тепер розглянемо конструкцію switch, загальний синтаксис наступний:

```
switch (вираз) {  
    case константа 1: послідовність інструкцій;  
    break;  
    case константа 2: послідовність інструкцій;  
    break;  
    case константа 3: послідовність інструкцій;  
    break;  
    default: послідовність інструкцій;  
}
```

Особливості використання switch:

- Інструкція switch відрізняється від інструкції if тим, що switch-вираз можна використовувати тільки з використанням умови рівності (тобто на збіг switch-виразу з однією з заданих case-констант), у той час, як умова if-виразу може бути будь-якого типу.
- Жодні дві case-константи в одній switch-інструкції не можуть мати ідентичні значення.
- Інструкція switch зазвичай більш ефективна, ніж вкладені if-інструкції.
- Послідовність інструкцій, пов'язана з кожною case-гілкою, не є блоком коду. Однак повна switch-інструкція визначає блок.

6.2. Завдання 1. Інтерфейс додатку «Макдональдс»

Створити програму, що буде реалізовувати інтерфейс користувача в ресторані типу фастфуд. Використовувати компонент CheckBox. Після обрання необхідних пунктів і натискання кнопки «ОК», з'являється повідомлення із загальною сумою у чеку. Якщо обрано всі позиції, надається знижка 5%. Якщо не обрано Біг-Мак, або картопля фрі, кетчуп не можна обрати. Загальний вигляд інтерфейсу показано на рис. 6.1.

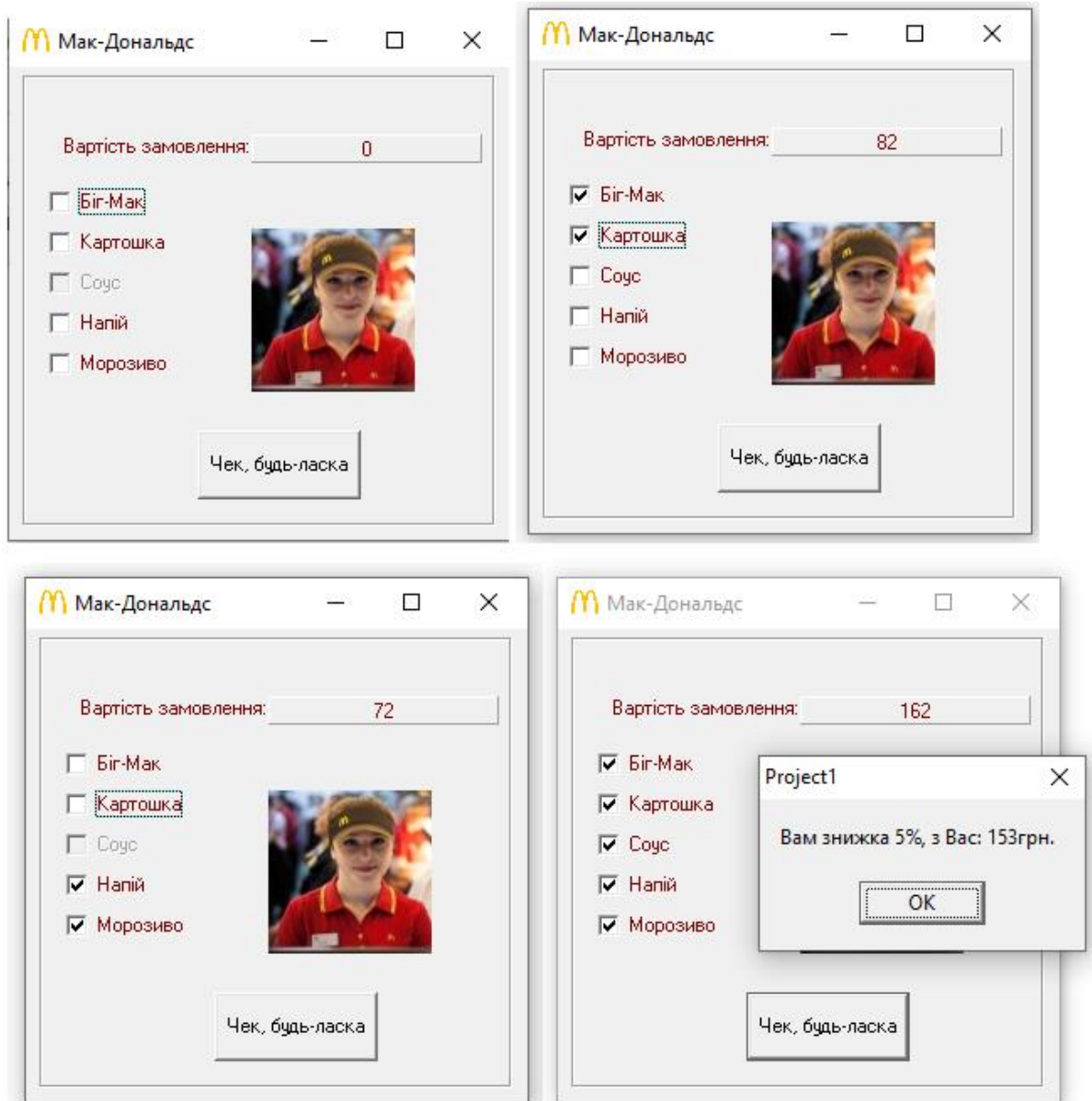


Рис. 6.1 – Загальний вигляд інтерфейсу додатку для завдання 1.

Для реалізації впливаючого вікна із загальним рахунком можна створити додаткову форму, як показано у попередньому комп'ютерному практикумі, або скористатись методом ShowMessage, наприклад:

ShowMessage("З Вас " + IntToStr(sum) + " грн.");

Також необхідно зауважити, що даний код достатньо реалізувати лише у вигляді VCL додатку (із графічним інтерфейсом). Консольний варіант може бути реалізований за власним бажанням здобувача на додатковий бал.

6.3. Завдання 2

Реалізувати код, що працює за наступним алгоритмом:

У трьох структурних підрозділах підприємства прибуток нараховується за наступними формулами:

- Для першого підрозділу α $y_1 = 100|f_i(i) + 100|$
- Для другого підрозділу β $y_2 = 120|f_{i+1}(i) + 120|$
- Для третього підрозділу γ $y_3 = 130|f_{i+2}(i) + 130|$,

де i - номер варіанта.

Для підрозділу α затрати на амортизацію становлять 20%, для β - 30% та для γ - 40%. Вивести нараховану розраховану суму, суму, що відраховується на амортизацію і суму прибутку для одного із обраних підрозділів.

Реалізувати код, використовуючи сходинкову конструкцію **if-else-if**, окремим додатком використовуючи **switch**.

Функцію f_i потрібно обрати із таблиці 6.1. Наприклад, для варіанту $i = 10$, якщо $y = f_{i+2}(x)$, то $i + 10 = 12$. Тож беремо формулу під номером 12 із зазначеної таблиці. Якщо отримали значення більше ніж 25, то починаємо спочатку, тобто $26=1, 27-2\dots$ т.д.

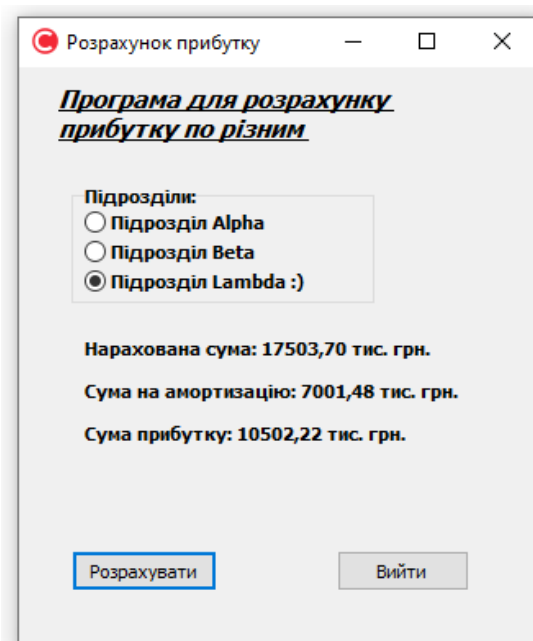


Рис. 6.1 – Загальний вигляд інтерфейсу додатку для завдання 2.

Таблиця. 6.1. Функції для завдання

№	Функція
1	$9,2\cos 2x - \sin x / 3,14 $
2	$12,4 \sin x/3,4 - 7,5 \cos 1,5x$
3	$ \cos x / 1,5 + 9,2 \sin (1,2x+5)$
4	$ \sin x / 3,14 + \cos x^2 - 10 \sin 3x$
5	$\cos 2x / 2,15 - \cos (3x-3) + 10$
6	$\sin x \sin(x+2) \cos^2 x / 0,44 + 11,4$
7	$ 3 \sin 3x + \sin (2x-1,8) + 3,45$
8	$4,31 + \cos^2 x \sin (2x-2)$
9	$\cos(x^3+1) - \sin 2x-5,5 $
10	$\sin x - \cos x^3 \sin (x^2-4,24) + 4,24$
11	$ \sin 5x \cos 2x /3 + 5,86$
12	$\cos x^3/2,4 + \cos x^2/2,4 - 9,46 \sin (3x+1)$
13	$\cos x^3 \sin x^2 - \sin x + 3,14$
14	$2\sin x \sin (4x-1,4)\cos (2x+1,5)-7,1$
15	$ \cos x^2 - 1,25 \sin (3x- 4,44)+4,44$
16	$\cos 3,4x \sin -x / 0,3 - 6,24$
17	$ \cos 2x^3 + \sin(14x-3) + 2,44$
18	$ \sin(x^2/1,5-1,5) + 11,4\cos(2x-2)$
19	$13,14\sin(-1,24x) \cos x-1,5 + 4,78$
20	$ \sin 3x \cos 5x / 1,4 + 4,44$
21	$\sin^2(x^2-1) - \sin 5x-4,5 + 2,12$
22	$2\sin 2x \cos 2x-12,4\sin(x/0,4-1)$
23	$10,4 \sin x^2 \sin x^3 - \sin x - 12,1$
24	$11,2 \cos(2x-1) + \sin 1,5x / 1,3$
25	$ 10 \sin(x^2/1,5-1,4) + 11,4\cos(2x-2) - 10$

Інтерфейс консольного додатку розробляється на власний розсуд, проте функціонал повинен бути збережений.

6.4. Зміст звіту

1. Код створених програм (перенести за допомогою Print Screen або скопіювавши). До коду бажано додавати коментарі, що пояснюють основні моменти і особливості виконання програми.

2. Фото із результатами виконання як в консолі, так і у формі.
3. Загальні висновки по роботі.

6.5. Контрольні запитання

1. Опишіть загальний формат інструкції if-else-if.
2. Чи повинна конструкція if-else-if мати гілку else? Обґрунтуйте із прикладом.
3. Чи вірно те, що умова, що керує if-інструкцією, повинна обов'язково використовувати оператор відношення?
4. Який загальний формат інструкції switch?
5. Якого типу повинен бути вираз, що керує інструкцією switch?
6. Що відбувається, якщо case-послідовність не завершується інструкцією break?
7. Наведіть приклади, в яких випадках ефективніше використовувати інструкцію break, а коли if.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 7

ЦИКЛІЧНІ АЛГОРИТМИ З ФІКСОВАНОЮ КІЛЬКІСТЮ ПОВТОРЕНЬ

Мета роботи: отримати навички роботи із інструкцією циклу `for` для реалізації циклічних алгоритмів.

7.1. Теоретичні відомості

Для виконання великої кількості однотипних дій або повторюваних частин коду на практиці використовують циклічні конструкції. Коли кількість ітерацій визначена, то у такому разі застосовують оператор `for`. Загальний синтаксис використання інструкції наступний:

for (початкова дія; умова виконання циклу; дія після ітерації)

інструкція ;

або так

for (лічильник = початкове значення; лічильник < кінцеве значення;

крок циклу)

{

інструкція 1;

інструкція 2;

...

інструкція N;

}

Для прикладу наведемо код, що знаходить суму значень від 1 до 10:

```
int i,sum = 0;
```

```
for(i=1; i <= 10; i++)
```

```
sum+=i;
```

```
cout << "Sum is " << sum;
```

7.2. Завдання 1. Переводи фізичних величин

Створити код із використанням циклу `for`, який виводить таблицю відповідності між значеннями фізичних величин. В якості початкових значень для розрахунку вводяться початкове значення величини, крок зміни та кількість рядків таблиці.

Варіанти завдань:

1. $1^{\circ}\text{C} = 33,8^{\circ}\text{F} = 274,15^{\circ}\text{K}$
2. $1 \text{ унція} = 28.353495 \text{ г} = 142 \text{ карати}$
3. $1 \text{ морська миля} = 6076 \text{ футів} = 1.852 \text{ км}$
4. $1 \text{ ярд} = 3 \text{ фути} = 0.9144 \text{ м}$
5. $1 \text{ стандарт} = 0.165 \text{ рода} = 4.672 \text{ куб. м}$
6. $1 \text{ парсек} = 3,086 \cdot 10^{13} \text{ км} = 1,92 \cdot 10^{13} \text{ миль}$
7. $1 \text{ драхм} = 1.77185 \text{ г} = 0.06249 \text{ унцій}$
8. $1 \text{ карат} = 2.9412 \text{ гран} = 0.2 \text{ г}$
9. $1 \text{ дюйм} = 2,54 \text{ см} = 0,083 \text{ фута}$
- 10.1 $\text{ пайп} = 54.18 \text{ пек} = 477.33 \text{ л};$
- 11.1 $\text{ секунда} = 0,0167 \text{ хв} = 2,78 \cdot 10^{-4} \text{ год}$
- 12.1 $\text{ род} = 28.3 \text{ куб. м} = 1000 \text{ куб. футів}$
- 13.1 $\text{ місяць} = 730 \text{ год} = 0,0833 \text{ року}$
- 14.1 $\text{ галон (США)} = 0.0347 \text{ сак} = 3.785 \text{ л}$
15. $1 \text{ Ккал} = 4184 \text{ Дж} = 2,778 \text{ КВт} \cdot \text{год}$
16. $1 \text{ градус} = 0,017453 \text{ рад} = 3600 \text{ кутових секунд}$
- 17.1 $\text{ чарка} = 0.0568 \text{ л} = 0.00012 \text{ пайпа}$
- 18.1 $\text{ квартет} = 5123.24 \text{ чарок} = 291 \text{ л}$
19. $1 \text{ атмосфера} = 1,01325 \text{ бар} = 101325 \text{ Па}$
- 20.1 $\text{ страйк} = 72.73 \text{ л} = 1280.46 \text{ чарок}$
- 21.1 $\text{ сак} = 1.499 \text{ страйк} = 109 \text{ л}$
- 22.1 $\text{ фінгер} = 11.4 \text{ см} = 4.5 \text{ дюймів}$
23. $1 \text{ Га} = 2,471 \text{ Акр} = 0,01 \text{ км}^2$
- 24.1 $\text{ нейл} = 2.25 \text{ дюймів} = 5.7 \text{ см}$
- 25.1 $\text{ фут} = 12 \text{ дюймів} = 0.3048 \text{ м}$

Увага! Завдання реалізовувати і в консольному додатку, і в формі!

Для виводу даних у вигляді таблиці, запропоновано використовувати об'єкт `Memo`. Для виведення написів необхідно використовувати, наприклад, наступний код:

```
Memo1->Lines->Add(" Унції \t Грами \t Карати");
```

Для того, щоб очистити поле `Memo` необхідно використати

```
Memo1->Clear();
```

Загальний вигляд інтерфейсу додатку у формі показано на рис. 7.1.

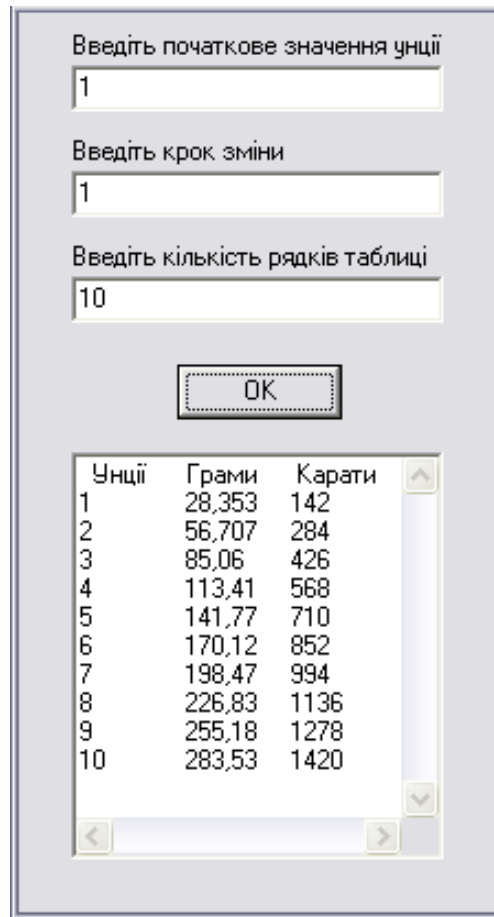


Рис. 7.1 – Загальний вигляд інтерфейсу додатку для завдання 1.

7.3. Завдання 2. Обрахунок сум та добутків

Використовуючи конструкцію циклу **for** розрахувати значення суми a і добутку b скінченних рядів, що задані формулами:

$$a = \sum_{k=1}^{i+8} f_{i+4}(k), \quad b = \prod_{k=i}^{i+4} f_{i+5}(k),$$

де i -номер варіанта, k -цілі числа. Функція f_i обрається із таблиці 6.1.

Після цього, обчислити значення змінної z для свого варіанта:

- | | |
|--------------------------|-----------------------|
| 1) $z = a + b$ | 2) $z = b - a$ |
| 3) $z = 100a - b$ | 4) $z = b / a$ |
| 5) $z = (0.1b)^a$ | 6) $z = (ab)^{1/4}$ |
| 7) $z = (a + b / 100)^2$ | 8) $z = \cos(ab)$ |
| 9) $z = 133a - b $ | 10) $z = \sin(a) + b$ |

- | | | | |
|-----|-------------------------|-----|--------------------------------------|
| 11) | $z = 4ab - b$ | 12) | $z = \ln 4a + b $ |
| 13) | $z = \cos(a) + \sin(b)$ | 14) | $z = a \sin(b)$ |
| 15) | $z = 100a - b $ | 16) | $z = \exp(b/a)$ |
| 17) | $z = a^{(b/100)}$ | 18) | $z = b - 12\pi a$ |
| 19) | $z = a^2 + 3b$ | 20) | $z = \operatorname{tg}(a) - \sin(b)$ |
| 21) | $z = \exp(a) - b$ | 22) | $z = \lg 5a - b $ |
| 23) | $z = a + 1/\exp(b)$ | 24) | $z = a\sqrt{b}$ |

Вивести значення i , a , b , z .

Для довідки. Символи \sum та \prod позначають скорочений запис суми та добутку ряду відповідно. Приклад повної форми запису:

$$\sum_{k=1}^3 a_k = a_1 + a_2 + a_3$$

$$\prod_{k=1}^3 a_k = a_1 * a_2 * a_3$$

Загальний вигляд інтерфейсу додатку у формі показано на рис. 7.2.

Завдання реалізовувати і в консольному додатку, і в формі!

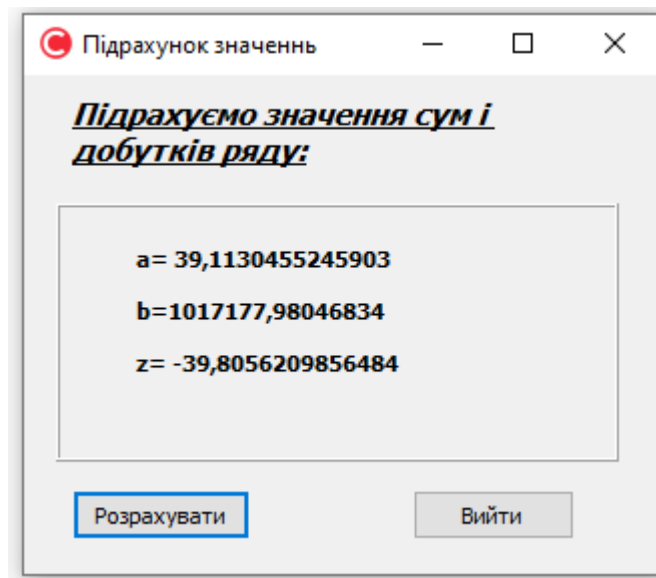


Рис. 7.2 – Загальний вигляд інтерфейсу додатку для завдання 2.

7.4. Зміст звіту

1. Код створених програм (перенести за допомогою Print Screen або скопіювавши). До коду бажано додавати коментарі, що пояснюють основні моменти і особливості виконання програми.
2. Фото із результатами виконання як в консолі, так і у формі.
3. Загальні висновки по роботі.

7.5. Контрольні запитання

1. Опишіть загальний формат інструкції `for`.
2. Як створити нескінченний цикл? Чому використання нескінченних циклів не рекомендується?
3. Які типи даних можуть бути у лічильника циклу, чому?
4. Як реалізувати код, що виведе на екран алфавіт від `A` до `Z`?
5. Чи може у дужках циклу бути відсутній один з елементів, наприклад початкове значення?
6. До чого призведе використання оператора `break` у тілі циклу?
7. До чого призведе використання оператора `continue` у тілі циклу?
8. Яка область видимості змінної, оголошеної в заголовку інструкції `for`?
9. Якого значення набуде змінна після виконання команд:
`a=5;`
`for (i=1;i<=2;i++) a=a*i-2; a++;?`

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 8

ЦИКЛИ ІЗ НЕВИЗНАЧЕНОЮ КІЛЬКІСТЮ ІТЕРАЦІЙ

Мета роботи: отримати навички роботи із інструкціями циклу `while` та `do-while` для реалізації циклічних алгоритмів.

8.1. Теоретичні відомості

У випадках, коли кількість повторень частини коду заздалегідь не визначена, або її важко спрогнозувати, замість конструкції циклу `for` використовують конструкцію циклу `while`. Загальний формат запису даної інструкції доволі простий і чимось нагадує інструкцію вибору `if`:

```
while (/*умова роботи циклу*/)  
    {  
        /*послідовність інструкцій*/;  
        /*керування умовою*/;  
    }
```

Інструкція перевіряє умову, що записана в круглих дужках, якщо значення, що повертає умова дорівнює **true** - виконується тіло циклу, якщо ні, відбувається вихід з циклу.

Для прикладу розглянемо програму, яка виводить значення від 1 до 10:

```
int i = 0;  
while (i < 10)  
    {  
        i++;  
        cout << i << "\t";  
    }
```

Не дивлячись на те, що кількість ітерацій тут визначена, даний приклад демонструє взаємозамінність інструкцій циклів `while` і `for`.

Аналізуючи наведений приклад легко помітити, що якщо умова не виконується, наприклад, якщо початкове значення `i` буде дорівнювати 15, то тіло циклу не виконається жодного разу. То у таких випадках, використовується модифікована конструкція `do-while`:

```
do // початок циклу do while
{
/* послідовність інструкцій */;
}
while (/* умова роботи циклу */);
```

Розглянемо програму, що завершує своє виконання після вводу літери “q”.

```
unsigned char ch;
do{
    cout<<"\n Вгадайте, яка кнопка завершує програму? \n";
    ch = getch();
}
while(ch!='q');
```

Тобто, основна відмінність від циклу while полягає у тому, що інструкції записані в do виконуються хоча б один раз.

8.2. Завдання 1. Обчислення суми нескінченного ряду

Обчислити суму ряду $\sum_{k=1}^{\infty} a_k$,

де $a_k = (-1)^k \frac{f_{i+3}(2k+1)x^k}{k!}$, i - номер варіанта, x - довільне значення з проміжку від 0 до 1. Функція f_i обирається з таблиці 6.1.

Програма повинна виводити значення суми ряду із заданою точністю та кількість доданків, що були підсумовані. Запишіть отримані значення у файл. Виконайте програму тричі для різних значень точності.

Зазначимо, що у прикладі наведено знакозмінний ряд, що сходиться. Так як обчислити нескінченну кількість доданків технічної можливості (і сенсу) немає, то сума такого ряду обчислюється із необхідною точністю. Для даного завдання прийmemo точність рівну з точністю $e = 0.001$. Основний принцип, чим більше доданків ряду, тим точніше обчислюється його сума.

Отже, щоб обчислити суму ряду із заданою точністю ϵ , необхідно порівнювати абсолютне значення кожного отриманого доданка із заданим значенням точності. Коли значення доданку по модулю стане меншим за ϵ – це і буде останній доданок, а отже рахувати переставо.

Запис у файл. Наступний шматок коду демонструє алгоритм запису у файл:

```
#include <fstream.h> // підключаємо заголовочний файл
ofstream fout; // створюємо об'єкт fout класу ofstream
fout.open("file.txt", ios::app); // відкриваємо файл "file.txt", якщо він не існує,
то файл буде автоматично створений.
```

Параметр `ios::app` вказує на те, що дані будуть додаватися у файл.

```
fout << "Це буде записане у файл"; // запис у файл
fout.close(); // закриття файлу
```

Зазначимо, що наведений код буде працездатний у консольних застосунках і у формі.

Загальний вигляд інтерфейсу додатку у формі показано на рис. 8.1.

Завдання реалізувати і в консольному додатку, і в формі!

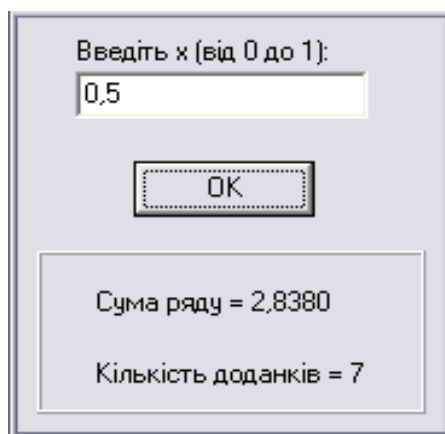


Рис. 8.1 – Загальний вигляд інтерфейсу додатку для завдання 1

Умовною перевіркою правильності підрахунку може бути збільшення точності, наприклад на два порядки $\epsilon = 0.00001$, у такому випадку кількість доданків має збільшитись, а значення суми ряду залишитись майже не змінним.

8.3. Завдання 2. Розрахунок функції за допомогою розкладання в ряд

Дана функція представлена у вигляді ряду Тейлора. Підрахувати значення заданої функції з уведеною точністю ε , та вивести кількість підсумованих членів ряду.

Провести підрахунок декілька разів від $x_{поч}$ до $x_{кін}$ з кроком dx .

$$1. \ln \frac{x+1}{x-1} = 2 \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = 2\left(\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \dots\right) \quad |x| > 1$$

$$2. e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \dots \quad |x| < \infty$$

$$3. e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \dots \quad |x| < \infty$$

$$4. \ln(x+1) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{n+1}}{n+1} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} - \dots \quad -1 < x \leq 1$$

$$5. \ln \frac{1+x}{1-x} = 2 \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \dots\right) \quad |x| < 1$$

$$6. \ln(1-x) = -\sum_{n=1}^{\infty} \frac{x^n}{n} = -\left(x + \frac{x^2}{2} + \frac{x^4}{4} + \dots\right) \quad -1 \leq x < 1$$

$$7. \operatorname{arcctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} x^{2n+1}}{2n+1} = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} - \dots \quad |x| \leq 1$$

$$8. \operatorname{arctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} \dots \quad x > 1$$

9. $\operatorname{arctg} x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots \quad |x| \leq 1$
10. $\operatorname{Arth} x = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \dots \quad |x| < 1$
11. $\operatorname{Arth} x = \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = \frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \dots \quad |x| > 1$
12. $\operatorname{arctg} x = -\frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \dots \quad x < -1$
13. $e^{-x^2} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots \quad |x| < \infty$
14. $\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots \quad |x| < \infty$
15. $\frac{\sin x}{x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n+1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} - \dots \quad |x| < \infty$
16. $\ln x = 2 \sum_{n=0}^{\infty} \frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}} = 2 \left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3} + \frac{(x-1)^5}{5(x+1)^5} + \dots \right) \quad x > 0$
17. $\ln x = \sum_{n=0}^{\infty} \frac{(-1)^n (x-1)^{n+1}}{(n+1)} = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} + \dots \quad 0 < x \leq 2$
18. $\ln x = \sum_{n=0}^{\infty} \frac{(x-1)^{n+1}}{(n+1)x^{n+1}} = \frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \frac{(x-1)^3}{3x^3} + \dots \quad x > \frac{1}{2}$
19. $\arcsin x = x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \dots \cdot (2n-1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \dots \cdot 2n \cdot (2n+1)} = x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} \dots \quad |x| < 1$
20. $\arccos x = \frac{\pi}{2} - \left(x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \dots \cdot (2n-1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \dots \cdot 2n \cdot (2n+1)} \right) =$
 $= \frac{\pi}{2} - \left(x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 8} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} \dots \right) \quad |x| < 1$

Загальний вигляд інтерфейсу додатку у формі показано на рис. 8.2.
 Завдання реалізовувати і в консольному додатку, і в формі!

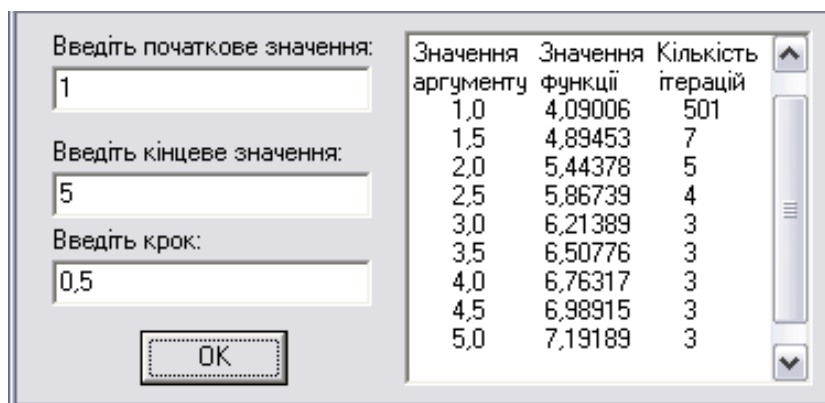


Рис. 8.2 – Загальний вигляд інтерфейсу додатку для завдання 2

Перевіркою правильності розрахунків може бути підрахунок із використанням функції, що записана ліворуч, наприклад: значення функції

$\cos(x)$ і значення суми ряду $1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$ мають бути майже однаковими при однаковому значенні аргументу x .

8.4. Завдання 3. Пошук даних

Дано функцію $y = f_{i+8}(x)$, її аргумент змінюється у діапазоні від $x_{поч}$ до $x_{кін}$ з кроком dx . Значення аргументу та відповідне значення функції вивести у вигляді таблиці пар чисел x та y .

Виконати завдання пошуку даних відповідно до вашого варіанта. Якщо таких значень не знайдено, вивести про це повідомлення, або змінити функцію (старий варіант функції залишити у коментарі).

Варіанти завдань:

1. Обчислити суму першого та останнього від'ємного значення функції. Визначити кількість усіх від'ємних значень.
2. Обчислити суму квадратів всіх від'ємних значень та їхню кількість.
3. Обчислити суму значень функції y , для яких виконуються нерівності $y < -5$ або $y < 0$.

4. Обчислити добуток усіх від'ємних значень функції y та визначити кількість додатних.
5. Обчислити суму всіх додатних значень функції y та визначити їх кількість.
6. Обчислити суму та кількість усіх значень функції y , для яких виконуються нерівності $y > -3$ і $y < 1$.
7. Обчислити добуток значень аргументів (x), при яких функція y приймає максимальне та мінімальне значення.
8. Обчислити середнє арифметичне всіх від'ємних значень функції та підрахувати кількість додатних значень.
9. Обчислити кількість і добуток аргументів, при яких функція y приймає від'ємні значення.
10. Визначити суму додатних значень функції та кількість від'ємних.
11. Обчислити середнє між мінімальним та максимальним значенням y , та середнє між аргументами при цих значеннях.
12. Скільки від'ємних і додатних значень має функція y ?
13. Обчислити суму від'ємних значень функції y . Яке із цих значень максимально наближене до 0?
14. Обчислити суму квадратів усіх додатних значень функцій y . Визначити, аргумент x для якого (із знайдених значень) функція набуває мінімального значення.
15. Обчислити суму та кількість додатних значень функції y .
16. Обчислити суму всіх значень функції y , для яких виконуються нерівності $y < 2$ або $y > 4$. Визначити максимальне значення функції.
17. Обчислити добуток додатних і добуток від'ємних значень.
18. Обчислити добуток усіх значень функції y , для яких справджується нерівність $1 < y < 3$. Визначити, для якого x функція набуває максимального значення.
19. Обчислити суму та кількість значень (x) при яких функція y приймала від'ємні значення.

20. Обчислити середнє значення аргументів при яких функція приймає максимальне і мінімальне додатне значення.

Загальний вигляд інтерфейсу додатку у формі показано на рис. 8.3.

Завдання реалізувати і в консольному додатку, і в формі!

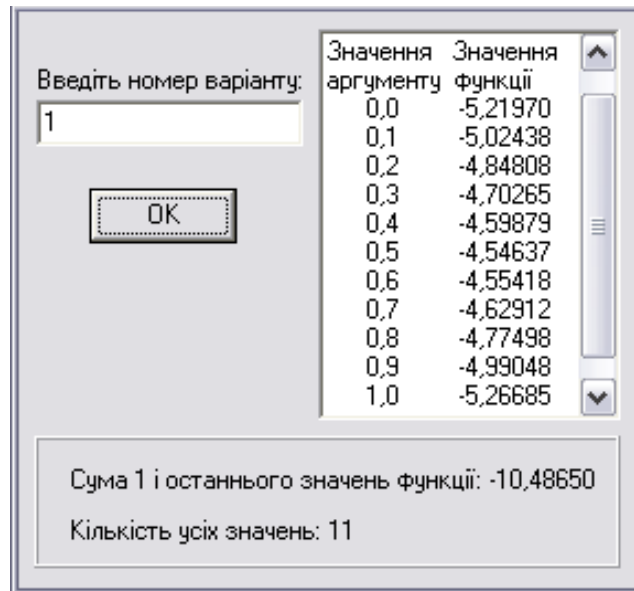


Рис. 8.3 – Загальний вигляд інтерфейсу додатку для завдання 3

8.5. Зміст звіту

1. Код створених програм (перенести за допомогою Print Screen або скопіювавши). До коду бажано додавати коментарі, що пояснюють основні моменти і особливості виконання програми.
2. Фото із результатами виконання як в консолі, так і у формі.
3. Загальні висновки по роботі.

8.6. Контрольні запитання

1. Опишіть загальний формат інструкції while.
2. Чи завжди цикли while і for взаємозамінні?
3. Чим відрізняються інструкції while і do-while?
4. Покажіть як за допомогою інструкції while створити нескінченний цикл, коли це потрібно, чи можна його, за необхідності, перервати?
5. Якого значення набуде змінна після виконання команд: `a=4; do a=-3*(a-6); while (a<0);`

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 9

МАСИВИ. ПОШУК ДАНИХ В ОДНОВИМІРНИХ МАСИВАХ

Мета роботи: навчитися створювати масиви та працювати з ними в C/C++, застосувати на практиці алгоритми пошуку даних в одновимірних масивах.

9.1. Теоретичні відомості

9.1.1. Масиви. Оголошення та робота з масивом

Масиви являються одним із видів структур даних, що містять зв'язані один з одним однотипні елементи даних. Класично масиви – це «статичні» об'єкти, які зберігають свої розміри при виконанні програми. Масиви використовуються для обробки великої кількості однотипних даних, які об'єднані одним ім'ям.

Для того, щоб звернутися до окремої комірки або елемента масиву потрібно вказати ім'я масиву та індекс окремого елемента масиву. У C++ перший елемент кожного масиву – це так званий «нульовий елемент», тобто елемент масиву з індексом 0. Загальна форма оголошення масиву має вигляд

<тип даних> Ім'я_масиву [розмір масиву];

Ім'я масиву має відповідати тим же самим вимогам, що й імена змінних. Наприклад, оголосимо масив цілих чисел, що містить 5 елементів, та масив чисел з плаваючою комою з 7 елементів:

```
const int IntArrSize = 5; // іменована константа, в яку записана  
кількість елементів масиву цілих чисел IntArr
```

```
const int FloatArrSize = 7; // іменована константа, в яку записана  
кількість елементів масиву чисел FltArr
```

```
int IntArr [ IntArrSize ];
```

```
float FltArr [ FloatArrSize ];
```

На рис 9.1 графічно продемонстровано розміщення елементів масивів, комірки яких заповненні довільними числами.

Порядковий номер елемента ⇒	1	2	3	4	5	6	7
Індекс елемента ⇒	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>
Масив IntArr ⇒	IntArr[0]	IntArr[0]	IntArr[0]	IntArr[0]	IntArr[0]		
значення ⇒	22	-5	12	0	2022		
Масив FltArr ⇒	FltArr[0]	FltArr[1]	FltArr[2]	FltArr[3]	FltArr[4]	FltArr[5]	FltArr[6]
значення ⇒	0,124	1,02	88,74	12,036	5,84	-1E-50	7,7E+8

Рис. 9.1 – Масиви з 5 та 7 елементів

Для того, щоб звернутися до певного елемента масиву треба написати ім'я масиву, а справа від нього у квадратних дужках написати індекс необхідного елемента. Зверніть увагу, що індекс елемента це не його порядковий номер, а значення на 1 менше (у зв'язку з тим, що індексація елементів масиву починається з 0). Квадратні дужки, всередині яких записується індекс масиву, у C++ розглядаються як операція індексації. Квадратні дужки мають той же рівень пріоритету, що й круглі дужки. Відповідно до вищевказаного, ім'я масиву є по суті вказівником на адресу в пам'яті, де розміщений перший елемент масиву.

При оголошенні масиву у його комірках можуть знаходитися будь-які числа, які були розміщені у тих комірках пам'яті, куди компілятор розмістить масив. А тому для визначення (за потреби) значень елементів масиву його можна ініціалізувати наступним чином:

<тип даних> Ім'я_масиву [розмір масиву] = { елемент 1, елемент 2, елемент N};

Ініціалізація може виконуватися лише при оголошенні масиву. Наприклад, ініціалізуємо попередній масив цілих чисел

const int IntArrSize = 5;

int IntArr [IntArrSize] = {0, -54, 12345, 6, 89};

Якщо при ініціалізації масиву було вказано недостатню кількість значень, що відповідає всім елементам масиву, то елементам, яким не вистачило значень ініціалізації, буде присвоєно 0 або відповідний аналог нуля. Продемонструємо це роботою наступного коду, у якому оголосимо три

масиви. Перший масив повністю ініціалізуємо, другий частково, а третій масив заповнимо випадковими значеннями за допомогою циклу **for**.

```
int _tmain(int argc, _TCHAR* argv[])
{
    SetConsoleCP(1251); SetConsoleOutputCP(1251);
    const int IntArrSize = 5; // іменована константа, в яку записана
    кількість елементів масиву цілих чисел IntArr
    const int FltArrSize = 7; // іменована константа, в яку записана
    кількість елементів масиву чисел FltArr
    const int CharArrSize = 10; // іменована константа, в яку записана
    кількість елементів масиву чисел FltArr
    int IntArr [ IntArrSize ]={-5, 0, 125, 86, 12};//повна ініціалізація
    float FltArr [ FltArrSize ]={1.22, 14587.457};//часткова ініціалізація
    char CharArr [ CharArrSize ];
    for (int i = 1; i <= CharArrSize; i++)
        CharArr[i-1] = 50+random(30);//заповнення символьного масиву (це не
    строка!) випадковими значеннями
    //виведення масивів в консоль з форматуванням
    cout << "Попередньо ініціалізований масив" << "\n";
    for (int i = 1; i <= IntArrSize; i++)
        cout << setw(5) << i << setw(10) << IntArr[i-1]<< endl;
    cout << "-----***-----" << "\n";
    cout << "Частково ініціалізований масив з елементами типу Float" <<
    "\n";
    for (int i = 1; i <= FltArrSize; i++)
        cout << setw(5) << i << setw(10) << FltArr[i-1]<< endl;

    cout << "_____*****_____" << "\n";
    cout << "Символьний масив, заповнений у циклі" << "\n";
    for (int i = 1; i <= CharArrSize; i++)
        cout << setw(3) << i << "-"<< setw(2) << CharArr[i-1]<<" ";
    cout<< endl;
```

```

    system("pause");
    return 0;
}

```

Результат роботи коду показано на рис.9.2.

```

Попередньо ініціалізований масив
1      -5
2       0
3     125
4     86
5     12
-----***-----
Частково ініціалізований масив з елементами типу Float
1     1.22
2   14587.5
3       0
4       0
5       0
6       0
7       0
*****
Символьний масив, заповнений у циклі
1- C;  2- @;  3- C;  4- M;  5- 8;  6- <;  7- A;  8- =;  9- F; 10- E;
Для продовження натисніть будь-яку клавішу . . .

```

Рис. 9.2 – Результати роботи програми, в якій було ініціалізовано та заповнено випадковим чином одновимірні масиви

Запам'ятайте, що у C++ компілятор не перевіряє правильність вказаного індексу. А тому із-за недосвідченості та необережності розробника можуть виникати ситуації звернення до неіснуючого елемента масиву, що виходить за розміри масиву. При зверненні програми до неіснуючого елемента компілятор не буде видавати помилок чи попереджень. Робота з неіснуючими елементами масиву може викликати проблеми під час виконання програми. Це може проявлятися у вигляді пошкодження даних, коду чи навіть аварійного завершення програми.

Як було сказано на початку практикуму, масиви зручно використовувати для обробки великої кількості однотипних даних. Розглянемо наступну задачу:

На 1-му потоці 100 студентів вивчали дисципліну «Психологія», а на 2-му потоці – 135. Вважаючи, що всі студенти отримали позитивні оцінки за 100-бальною шкалою, потрібно знайти середнє арифметичне та середнє геометричне значення оцінки на кожному з потоків.

У лістингу програми, що демонструє застосування масивів для вирішення цієї задачі, для кожного потоку оголошений свій масив. Кількість елементів масиву задана іменованою константою, що дозволяє дуже просто змінювати кількість елементів, які оброблятиме програма. Оцінки студентів кожного потоку задаються у відповідних циклах випадковим чином. Після визначення оцінки *i*-го студента виконується ітеративне знаходження суми та добутку оцінок потоку. Початкове значення змінної, що містить суму дорівнює 0. А змінної, що містить добуток – 1, щоб не обнулити відразу весь результат. Ця змінна має тип даних **double**, оскільки перемноження оцінок дуже швидко може привести до переповнення цілочисельного типу даних. Після завершення циклу реалізується розрахунок середньоарифметичного та середньгеометричного значень оцінки потоку.

```
int _tmain(int argc, _TCHAR* argv[])
{
    SetConsoleCP(1251); SetConsoleOutputCP(1251);
    randomize();//ініціалізація лічильника випадкових чисел

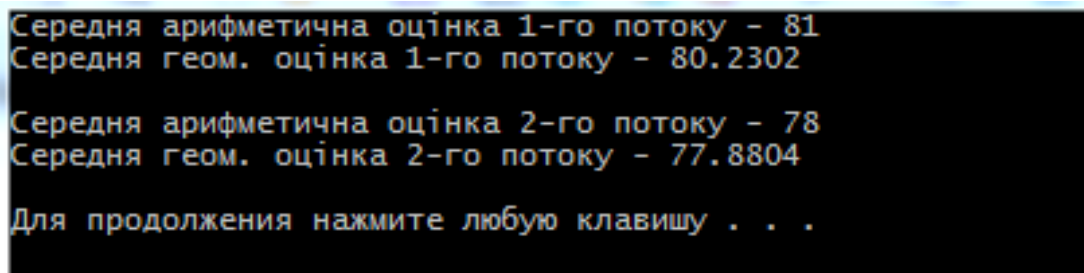
    const int FirstStreamSize = 100; //к-сть студентів 1-го потоку
    const int SecondStreamSize = 135; //к-сть студентів 2-го потоку
    int Stream1[FirstStreamSize], Stream2[SecondStreamSize];//оголошення
масивів, що міститимуть оцінки відповідного студента
    int sum1=0;
    double prod1=1;
    for(int i=1; i<=FirstStreamSize; i++)
    {
        Stream1[i-1] = 60 + random(40);// оцінки 1-го потоку
        sum1+= Stream1[i-1];
        prod1*= Stream1[i-1];
    }
    float aver_ar1 =sum1/FirstStreamSize;//сеп. арифметичне
    float aver_geom1 = pow(prod1, 1.0/FirstStreamSize);// сеп. геометричне
    int sum2=0;
```

```

double prod2=1;
for(int i=1; i<=SecondStreamSize; i++)
{
    Stream2[i-1] = 60 + random(40); // оцінки 2-го потоку
    sum2+= Stream2[i-1];
    prod2*= Stream2[i-1];
}
float aver_ar2 =sum2/SecondStreamSize;
float aver_geom2 = pow(prod2, 1.0/SecondStreamSize);
cout << "Середня арифметична оцінка 1-го потоку - " << aver_ar1
<< endl << "Середня геом. оцінка 1-го потоку - " << aver_geom1 << "\n\n";
cout << "Середня арифметична оцінка 2-го потоку - " << aver_ar2
<< endl << "Середня геом. оцінка 2-го потоку - " << aver_geom2 << "\n\n";
    system("pause");
    return 0;
}

```

Результат роботи коду показано на рис.9.3.



```

Середня арифметична оцінка 1-го потоку - 81
Середня геом. оцінка 1-го потоку - 80.2302

Середня арифметична оцінка 2-го потоку - 78
Середня геом. оцінка 2-го потоку - 77.8804

Для продовження натисніть будь-яку клавішу . . .

```

Рис. 9.3 – Обчислення результату успішності двох потоків студентів з дисципліни «Психологія»

9.1.2. Алгоритм пошуку

При роботі з великими масивами даних програмісту може бути потрібно визначити чи міститься у масиві елемент, значення котрого відповідає певному значенню чи умові – *ключовому значенню*. Це і є процес пошуку в масиві серед його елементів. У даному підрозділі буде розглянуто і описано базові та найбільш розповсюдженні алгоритми пошуку:

- 1) Лінійний пошук.

2) Двійковий пошук (дихотомія).

3) Інтерполяційний пошук.

Лінійний пошук. Алгоритм лінійного пошуку послідовно порівнює кожен елемент з ключем пошуку (з шуканим значенням). Якщо масив не впорядкований, то шуканий елемент може бути як першим, так і останнім. Але в середньому програма має виконати порівняння з ключем половини елементів масиву. Даний алгоритм добре працює для невеликих чи невідсортованих масивів. Для великих, а особливо відсортованих масивів, лінійний пошук неефективний.

Двійковий пошук. У відсортованому по певному параметру масиві більш раціонально використовувати ефективніші алгоритми, як наприклад, двійковий пошук. Цей алгоритм виключає половину ще неперевіраних елементів масиву після чергового порівняння. Алгоритм визначає розташування середнього елемента і порівнює його з ключем пошуку. Якщо значення рівні, то пошук завершено і видається індекс визначеного елемента. Якщо – ні, то задача пошуку скорочується вдвічі: якщо ключ пошуку менший за середній елемент масиву, то далі пошук в першій половині масиву. Якщо ключ більше – то пошук продовжується в другій половині масиву. У обраному підмасиві знову виконується порівняння з середнім елементом і повторно застосовується вищеописана процедура, доки не буде знайдено шукане значення (див.рис.9.4).

		ind 0	ind 1	ind 2	ind 3	ind 4	ind 5	ind 6	ind 7	ind 8	ind 9	ind 10	ind 11	
I war	///	1	2	3	4	5	6	7	8	9	10	11	12	///
		left					midd							right
II war	///	1	2	3	4	5	///	///	///	///	///	///	///	///
		left		midd		right								
III war	///	///	///	///	4	5	///	///	///	///	///	///	///	///
					left / midd	right								

Рис. 9.4 – Ілюстрація до алгоритму двійкового пошуку

Інтерполяційний пошук. Пошук, також як і бінарний, працює з відсортованими масивами, послідовність в яких прямує до арифметичної прогресії.

Пошук відбувається подібно до двійкового пошуку, але замість поділу області пошуку на дві приблизно рівні частини, інтерполяційний пошук проводить оцінку нової області пошуку по відстані між ключем і поточним значенням елемента.

9.1.3. Приклад пошуку за складним ключем

Тепер, розглянемо практичну задачу, в якій виконаємо пошук за складним ключем. Одразу зазначимо, що пошук будемо проводити із використанням найпростішого, лінійного пошуку:

Потік, що складається з 85 студентів, вивчав дисципліну «Вища математика». Припускаючи, що всі студенти отримали позитивні оцінки за 100-бальною шкалою, потрібно підрахувати скільки оцінок тої чи іншої категорії отримали студенти. Знайти кількість студентів, що мають бал вищий за середній та вивести в консоль порядковий номер елемента масиву, в якому записано бал вищий за середній на 5 балів і цей елемент є 4 для останньої умови.

Програма, що розв'язує цю задачу має виконати наступні елементи:

1) Розрахунок середнього арифметичного балу.
2) Класифікація оцінок за університетською шкалою. Будемо користуватися шкалою КПІ ім. Ігоря Сікорського (60-64 бали – Достатньо; 65-74 – Задовільно; 75-84 – Добре; 85-94 – Дуже добре; 95-100 – Відмінно). Для цього введемо масив з 5 елементів, в якому накопичуватимемо кількість відповідних оцінок.

3) Виконати пошук за ключем середнього балу, який вище середнього, і для таких елементів масиву знайти порядковий номер елемент масиву, що буде 4 для першого ключа пошуку. Використовуватимемо найпростіший лінійний алгоритм пошуку.

```
int _tmain(int argc, _TCHAR* argv[]){  
SetConsoleCP(1251); SetConsoleOutputCP(1251);  
randomize();//ініціалізація лічильника випадкових чисел
```



```

ofstream fout;      // потік для запису у файл
fout.open("Marks_Data.txt"); // відкриваємо файл для запису
const int StreamSize = 85; // к-сть студентів на потоці
int Stream1[StreamSize]; // оголошення масива, що містить оцінки
відповідного студента
int MarkTypeCount[5]={0};
if (fout.is_open())
    fout << "Student Marks" << endl;
    int sum1=0;
for(int i=1; i<=StreamSize; i++)
{
    Stream1[i-1] = 60 + random(40); // оцінки 1-го потоку
    sum1+= Stream1[i-1];
    if (fout.is_open())
        fout << Stream1[i-1] << "; " << endl;
}
float aver_ar1 =sum1/StreamSize; //середнє арифметичне
cout << "Середня арифметична оцінка потоку з ВМ- " << aver_ar1 <<
"\n\n";
//Пошук за складним ключем
int count_for_key = 0; // змінна, яка рахує порядковий номер елемента,
// для якого виконується умова, що його рейтинг більше
// на 5 балів за середній
int pos_for_key = -1; // змінна, в якій запам'ятовується порядковий номер
елемента
// масиву для якого виконані умови пошуку
int count Stud = 0; // змінна, в якій рахується кількість студентів з балом
вищій за середній

for(int i=1; i<=StreamSize; i++)
{
    if (Stream1[i-1]>=95) //розподіл оцінок
        MarkTypeCount[4]++;
    else if (Stream1[i-1]>=85)
        MarkTypeCount[3]++;
    else if (Stream1[i-1]>=75)
        MarkTypeCount[2]++;
    else if (Stream1[i-1]>=65)
        MarkTypeCount[1]++;
}

```

```

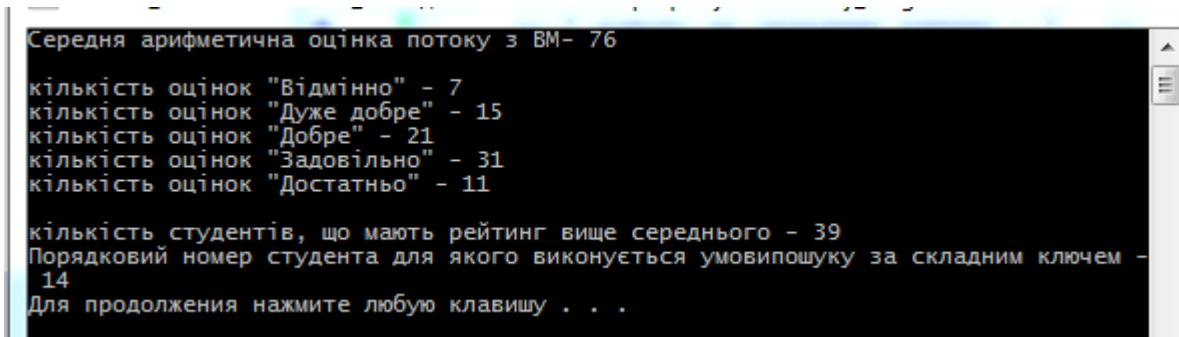
else
    MarkTypeCount[0]++;
//пошук за складним ключем
if (Stream1[i-1]>aver_ar1+5)
    count_for_key++;
if (Stream1[i-1]>(aver_ar1+5) && 4==count_for_key )
    pos_for_key = i-1;
    if (Stream1[i-1]>aver_ar1) //Підрахунок к-сті студентів з
балом вище за середній
        count_stud++;
}
cout << "кількість оцінок \"Відмінно\" - " << MarkTypeCount[4] <<
endl << "кількість оцінок \"Дуже добре\" - " << MarkTypeCount[3] <<
endl << "кількість оцінок \"Добре\" - " << MarkTypeCount[2] <<
endl << "кількість оцінок \"Задовільно\" - " << MarkTypeCount[1] <<
endl << "кількість оцінок \"Достатньо\" - " << MarkTypeCount[0] <<
endl << endl;
cout << "кількість студентів, що мають рейтинг вище середнього - "
<< count_stud << endl;
cout << "Порядковий номер студента для якого виконується умови"
<< "пошуку за складним ключем - " << pos_for_key+1 << endl;
if (fout.is_open()) {
    fout << "кількість оцінок \"Відмінно\" - " << MarkTypeCount[4] <<
endl << "кількість оцінок \"Дуже добре\" - " << MarkTypeCount[3] <<
endl << "кількість оцінок \"Добре\" - " << MarkTypeCount[2] <<
endl << "кількість оцінок \"Задовільно\" - " << MarkTypeCount[1] <<
endl << "кількість оцінок \"Достатньо\" - " << MarkTypeCount[0] <<
endl << endl;

    fout << "кількість студентів, що мають рейтинг вище середнього - "
<< count_stud << endl;
    fout << "Порядковий номер студента для якого виконується умови"
<< " пошуку за складним ключем - " << pos_for_key+1 <<
endl;

    fout.close();
}
system("pause");
return 0;}

```

Результат роботи коду показано на рис. 9.4. Також у підкаталозі проекту «Win32\Debug\» є виконуваний файл і текстовий файл «Marks_Data.txt», в який записано всі дані роботи програми.



```
Середня арифметична оцінка потоку з ВМ- 76
кількість оцінок "Відмінно" - 7
кількість оцінок "Дуже добре" - 15
кількість оцінок "Добре" - 21
кількість оцінок "Задовільно" - 31
кількість оцінок "Достатньо" - 11

кількість студентів, що мають рейтинг вище середнього - 39
Порядковий номер студента для якого виконується умовипошуку за складним ключем -
14
Для продовження натисніть будь-яку клавішу . . .
```

Рис. 9.4 – Лінійний пошук в масиві за складним ключем

9.2. Завдання 1. Одновимірні масиви

Вважаючи, що продуктивність фабрики з виробництва мікроелектронних чіпів та мікросистемних приладів за k -й рік розраховується за формулою $y_k = 100 f_{i+9}(k)$ тисяч виробів, де k - це роки починаючи з 2011р. по 2022р., i - номер варіанта. Кількість вироблених приладів по роках міститься в одновимірному масиві. Якщо $y_k > 0$, то вважається, що фабрика певного року мала прибуток за вироблену продукцію, а у випадку $y_k < 0$ — збитки, у зв'язку з тим що нереалізована продукція залишилась на складі. Вивести на екран таблицю: номер року, кількість реалізованої продукції.

Пошук даних. Розглянути діяльність фабрики протягом десяти років. Виконати додатково індивідуальне завдання вашого варіанта, відповідно до нижченаведеного переліку. Вивести повідомлення, якщо шуканих даних немає, наприклад, якщо якогось року взагалі нічого не було вироблено.

1. Обчислити суму кількості виробів, у ті роки коли фабрика мала прибутки. Визначити максимальний збиток по кількості нереалізованих виробів (якщо збитки були).
2. Обчислити суму кількості виробів для збиткових років. У якому році збиток по кількості нереалізованих виробів був максимальний?

3. Обчислити суми кількості виробів прибуткових і збиткових років фабрики та їх різницю. Коли було вироблено максимальну кількість виробів?
4. Скільки років поспіль кількості виробів у прибуткових роках була менше, ніж 1000, але більше, ніж 500? Коли фабрика зазнала найбільших збитків?
5. Обчислити суму кількості приладів для збиткових років. У якому прибутковому році було вироблено найбільше приладів?
6. Обчислити суму кількості приладів у межах $0 < y_k < 710$ (в тисячах виробів). У якому збитковому році фабрика зазнала найбільших збитків?
7. Скільки років виробництво приладів були в межах від 200 до 700 тисяч виробів? Які це були роки?
8. Обчислити суму кількості приладів для всіх збиткових років. У якому збитковому році фабрика зазнала найбільших збитків? Скільки приладів було вироблено у цьому році?
9. Обчислити суму кількості приладів по збитковим рокам, для яких справджуються умови $y_k < -550$ або $y_k > -150$ (тисяч виробів). Визначити найбільшу кількість вироблених приладів.
10. Визначити суми кількості приладів для прибуткових і збиткових років. Скільки років фабрика була прибутковою?
11. Обчислити суму кількості приладів по прибутковим рокам, у яких кількість виробів була у межах $230 < y_k < 8500$ (тисяч виробів). Скільки років фірма мала такі прибутки?
12. Обчислити суму кількості приладів для збиткових років, у яких кількість виробів була у межах $-750 < y_k < -200$ (тисяч виробів). У якому прибутковому році було вироблену найменшу кількість приладів?
13. Обчислити суму кількості приладів для прибуткових та збиткових років протягом перших семи років роботи фабрики, знайти їх різницю.

Визначити максимальну кількість приладів для прибуткових років за цей період.

14. Обчислити суми одиниць приладів для прибуткових років, у яких кількість виробів була у межах $y_k < 170$ або $y_k > 620$ (тисяч виробів). Скільки років фабрика мала таку кількість виробів?
15. Обчислити суму кількості приладів для збиткових років і визначити скільки років фабрика була збитковою? У якому збитковому році кількість приладів була максимальною?
16. Визначити кількість не реалізованих приладів для найбільш збиткового року. У якому прибутковому році фабрика виробила найбільшу кількість приладів?
17. У які роки фабрика мала найбільші прибуток і збиток?
18. Обчислити суму кількості приладів для збиткових років. Чи були такі роки, в яких не було вироблено жодного приладу?
19. Обчислити суму кількості приладів для прибуткових та збиткових років, знайти їх різницю. Визначити максимальну кількість приладів, що залишилися на складах (для збиткових років).
20. Обчислити суму кількості приладів для збиткових років, у яких для кількості виробів справджується умова $y_k < -590$ або $y_k > -330$ (тисяч виробів). Визначити кількість приладів для найбільш прибуткового року і вивести цей рік.

Приклад загального вигляду графічного вікна програми зображено на рис. 9.5.

9.3. Завдання 2. Одновимірні масиви та пошук даних за ключем

Утворити і вивести масив y з елементами $y_k = f_{i+10}(k)$, де i - номер варіанта, $k = 1, 2, \dots, 10$. Виконати завдання вашого варіанта. У разі відсутності шуканих даних вивести про це повідомлення.

1. Перший додатний елемент поміняти місцями з максимальним.
2. Знайти суму третього та шостого додатних елементів.
3. Другий від'ємний елемент замінити мінімальним.

4. Скільки є елементів з мінімальним значенням серед додатних?
5. Усі додатні елементи масиву, крім максимального, занести в інший масив.

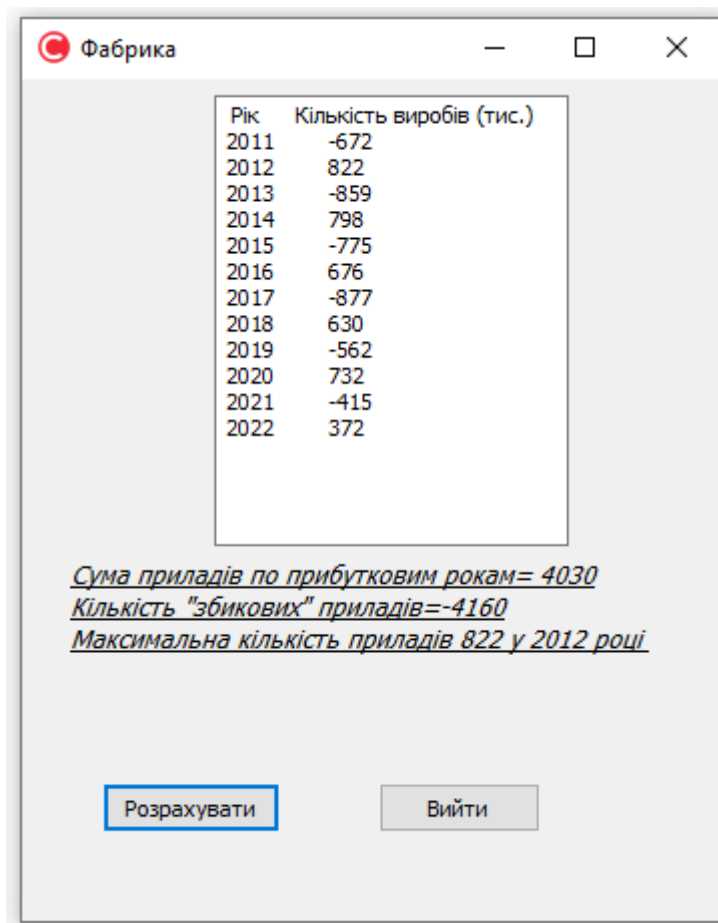


Рис. 9.5 – Загальний вигляд інтерфейсу додатку для завдання 1.

6. Обчислити суму перших чотирьох від'ємних елементів.
7. Вивести номер передостаннього додатного елемента.
8. Елементи масиву після другого від'ємного занести в інший масив.
9. Знайти добуток другого та четвертого елементів, більших, ніж 3.
10. Максимальний елемент поміняти місцями з другим нульовим.
11. Останній від'ємний елемент замінити найбільшим.
12. Обчислити добуток другого від'ємного та п'ятого елементів.
13. Чи третій додатний елемент є останнім у масиві?
14. Вивести номери двох найбільших елементів. Обчислити їх суму.
15. Чи є два елементи серед від'ємних із максимальним значенням?

16. Максимальний елемент поміняти місцями з четвертим, що задовольняє умову $y_k > 1$.

17. Третій додатний елемент замінити максимальним.

18. Визначити номер п'ятого від'ємного елемента.

19. Обчислити добуток перших трьох додатних елементів та визначити їхні номери.

20. Обчислити суму другого додатного та третього елементів.

Приклад загального вигляду графічного вікна програми зображено на рис. 9.6.

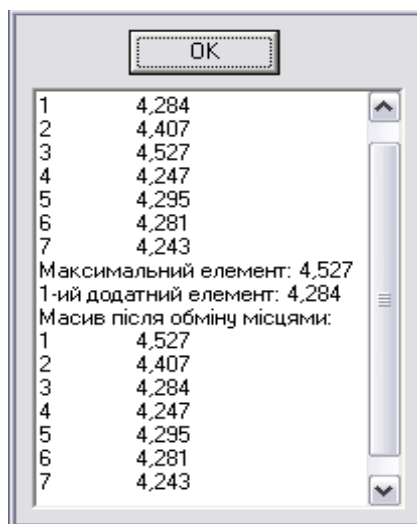


Рис. 9.6 – Загальний вигляд інтерфейсу додатку для завдання 2.

9.4. Зміст звіту

1. Код створених програм (перенести за допомогою Print Screen або скопіювавши). До коду бажано додавати коментарі, що пояснюють основні моменти і особливості виконання програми.

2. Фото із результатами виконання як в консолі, так і у формі.

3. Загальні висновки по роботі.

9.5. Контрольні запитання

1. Що таке масив? Які типи даних можна використовувати для створення масивів?

2. З якого числа починається індексація елементів масивів в C++? Чи можна це змінити?
3. Як оголошується масив? Як ініціалізувати масив початковими значеннями? Загальна форма оголошення масиву.
4. Що буде якщо задати недостатню кількість значень для ініціалізації всіх елементів масиву?
5. Що таке операція індексації і як вона пов'язана з масивами? Якими знаками задається ця операція? Який порядок пріоритету вона має?
6. Як звернутися до певного елемента масиву? Як звернутися до 100-го елемента масиву?
7. Які алгоритми пошуку ви знаєте? Що таке лінійний пошук?
8. Як виконується двійковий пошук?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 10

ПОШУК ДАНИХ В БАГАТОВИМІРНИХ МАСИВАХ

Мета роботи: удосконалити навички роботи з масивами та пошуку даних в них, навчитися працювати з візуальними елементами для роботи з масивами.

10.1. Теоретичні відомості

10.1.1. Багатовимірні масиви

Іноді отриману інформацію зручно розташовувати у вигляді таблиць, у такому випадку, у межах синтаксису мови C++ має сенс використовувати двовимірні масиви. Оголошення такого масиву відбувається за наступним синтаксисом:

тип ім'я_масиву [кількість рядків] [кількість стовпців];

Для більш детального пояснення наведемо приклад ініціалізації двовимірного масиву:

```
int massiv_2dim[4][2]= {    {1    , 4  },
                           {8    , -8  },
                           {114  , 0   },
                           {-9   , -1  } };
```

Основні підходи роботи із двовимірними масивами залишаються такими самими, як і для одновимірних з урахуванням додаткового індексу. Для прикладу наведемо код, що знаходить максимальне значення масиву розмірністю 4x6 заповненого випадковими значеннями:

```
randomize();
int array_2_dim[4][6];
int i,j,min=0,max=0,ind_i,ind_j;
for (i=0;i<4;i++)
{
for ( j=0; j<6; j++)
array_2_dim[i][j]=random(101);
}
for (i=0;i<4;i++)
{
```

```

        for ( j=0; j<6; j++)
if (max<array_2_dim[i][j])
    {
        max=array_2_dim[i][j];
        ind_i=i;
        ind_j=j;
    }
    }
    /// Виведення масиву
for ( i = 0; i < 4; i++) {
    cout<< endl;
    for ( j = 0; j < 6; j++)
        cout << "["<<i<<"]"<< "["<<j<<"]"<< " = "
<< array_2_dim[i][j] << "\t";
    }
    cout<<endl;
    cout<<"MAX = "<< "["<<ind_i<<"]"<< "["<<ind_j
<<"]"<< " = "<<max;
    cout<<endl;

```

Результат роботи коду показаний на рис. 10.1.

```

[0][0] = 67    [0][1] = 14    [0][2] = 34    [0][3] = 34    [0][4] = 82    [0][5] = 71
[1][0] = 96    [1][1] = 14    [1][2] = 59    [1][3] = 83    [1][4] = 39    [1][5] = 92
[2][0] = 3     [2][1] = 25    [2][2] = 32    [2][3] = 45    [2][4] = 74    [2][5] = 92
[3][0] = 90    [3][1] = 15    [3][2] = 48    [3][3] = 21    [3][4] = 30    [3][5] = 46
MAX = [1][0] = 96

```

Рис. 10.1 – Знаходження максимального значення у двовимірному масиві

Масиви більшої розмірності використовуються не так часто, як одна- та двовимірні масиви, проте принцип оголошення, ініціалізації і роботи з такими масивами залишається незмінним. Для пояснення наведемо приклад ініціалізації тривимірного масиву розмірністю 3x3x2:

```

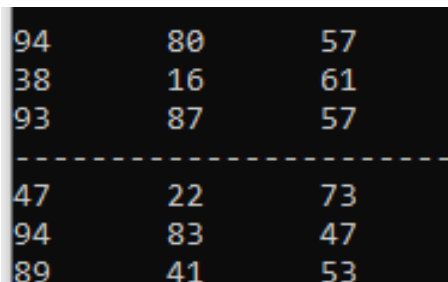
int massiv_3dim[3][3][2]= {{ {1 , 4},
                             {8 , -8},
                             {114, 0} },
                           { {1 , 4},
                             {8 , -8},
                             {114, 0} },
                           { {1 , 4},
                             {8 , -8},
                             {114, 0} }};

```

Тепер розглянемо наступний код і результат його виконання, що показаний на рис. 10.2:

```
randomize();
int array_2_dim[3][3], int array_1_dim[9];
int i,j;
for (i=0;i<3;i++)
{
for (j=0;j<3;j++) // формуємо одновимірний масив
array_2_dim[i][j]=random(101) ;
}
for (i = 0; i < 3; i++) { // формуємо двовимірний масив
cout<< endl;
for ( j = 0; j < 3; j++)
cout << array_2_dim[i][j] << "\t";
}
cout<<endl;
//-----
cout<<"-----"<<endl;
for ( j=0; j<9; j++)
array_1_dim[j]=random(101) ;
for ( j=0; j<9; j++)
{
if (j==3 || j==6)
cout<<endl;

cout <<array_1_dim[j]<< "\t";
}
cout<<endl;
```



```
94      80      57
38      16      61
93      87      57
-----
47      22      73
94      83      47
89      41      53
```

Рис. 10.2 – Результат виконання коду, що виводить два масиви

Якщо уважно розглянути вищенаведений код, то можна побачити, що формується два різні масиви `int array_2_dim[3][3]`, `int array_1_dim[9]`; двовимірний та одновимірний відповідно, але із однаковою кількістю комірок. При виведенні, візуально вони виглядають як два двовимірні масиви однакової розмірності. Це демонструє взаємозамінність одно та багатовимірних масивів, та відносність таких конструкції. Таким прийомом не варто користуватись, коли інформація, що зберігається у рядках/стовпцях двовимірних масивів принципово неоднорідна.

10.1.2. Компонент **StringGrid**

StringGrid – це візуальний компонент для роботи з масивами у вигляді таблиці, що може відображати як текстову, так і графічну інформацію. Вводити текст можна програмно в окремі комірки, або відразу по стовпчиках і рядках.

Наприклад, код:

```
StringGrid1->Cells[0][0]="Снівробітники";  
for(int r=1;r<=15;r++) StringGrid1->Cells[0][r]=IntToStr(r);
```

виконує заповнення першого стовпчика таблиці як показано на рис. 10.3.

Розглянемо основні властивості компонента. Властивості **ColCount** і **RowCount** визначають відповідно число стовпців і рядків.

Властивість **ScrollBars** визначає наявність у таблиці смуг прокрутки. Вони з'являються і зникають автоматично залежно від того, поміститься таблиця у задані розміри або ні.

При додаванні смуг прокрутки є можливість задати число рядків і стовпців, що будуть нерухомими і не будуть прокручуватися. Так можна задати заголовки стовпців і рядків, що будуть постійно зафіксовані в таблиці. Тож, властивості **FixedCols** та **FixedRows** визначають число фіксованих стовпців і рядків, які не прокручуються.

Також, за допомогою властивості **FixedColor** можна призначити кольори фону фіксованих комірок.

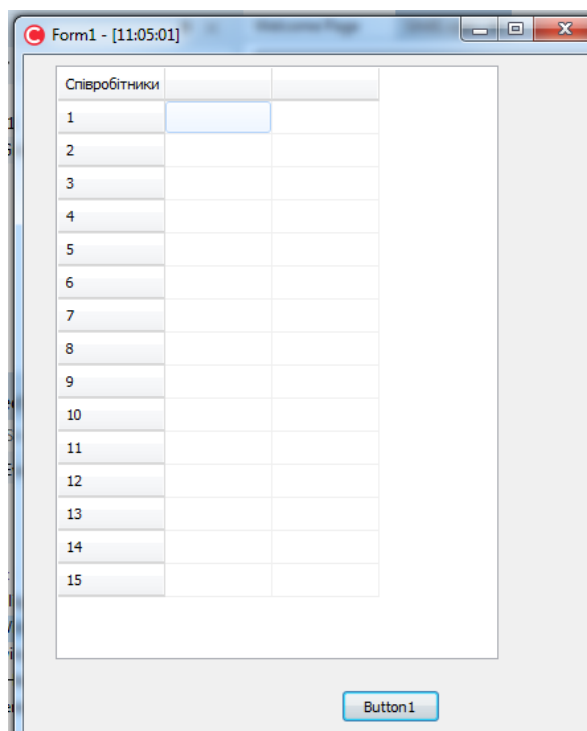


Рис 10.3 – Заповнення 1-го стовпчика у компоненті **StringGrid1**

Options – властивість-множина, що задає декілька властивостей таблиці: наявність розділяючих вертикальних і горизонтальних ліній у фіксованих комірках (**goFixedVertLine** та **goFixedHorzLine**) і не фіксованих комірках (**goVertLine** та **goHorzLine**); чи може користувач змінювати розміри стовпців і рядків, використовуючи мишку (**goColSizing** й **goRowSizing**), пересувати стовпці та рядки (**goColMoving** та **goRowMoving**) і т.п. Редагувати вміст таблиці дозволяє вибір елемента **goEditing** у властивості **Options**.

10.2 Завдання 1. Масиви із різною кількістю елементів

У підрозділі А є 15 співробітників, а в В - 20. Протягом місяця вони відпрацювали певну кількість днів, яка задана як випадкове число зі значенням від 0 до 31. Денна оплата праці 40 у.о. Податкова ставка 20%. Утворити двовимірні масиви А, В, оформити отримані таблиці якнайкраще (співробітник, дні, зарплата, податок), отримані данні зберегти у файл. Виконати завдання пошуку даних для кожного підрозділу. Якщо шуканих даних немає – вивести про це повідомлення.

Варіанти завдань:

1. Скільки осіб працювали у кожному підрозділі більше 15 днів?
2. Кому з робітників нарахували більше ніж 200 у.о.?
3. Хто з робітників заробив найменше у кожному підрозділі?
4. Знайти середній заробіток у кожному підрозділі?
5. Скільки людино-днів було відпрацьовано у кожному підрозділі?
6. Скільки осіб отримали від 300 до 500 у.о. в кожному підрозділі?
7. Скільки осіб працювали від 10 до 20 днів?
8. Який підрозділ сплатив більший податок?
9. Хто з робітників сплатив найбільший податок?
10. Скільки робітників відрахували на податки більше 100 у.о.?
11. У якому підрозділі більший середній заробіток?
12. Який середній податок був у кожному підрозділі?
13. Хто сплатив найменший податок у кожному підрозділі?
14. Скільки робітників працювало менше 10 днів?
15. Скільки робітників отримали заробіток вищий за середній?
16. Який підрозділ відпрацював більше людино-годин?
17. Скільки робітників отримали від 200 до 600 у.о. по двох підрозділах сумарно?
18. У якому підрозділі був зафіксований найбільший заробіток?
19. Скільки осіб працювали більше всього, менше ніж 10 днів, або більше 25 днів?
20. Хто із перших 10 робітників отримав найбільшу заробітну плату у кожному підрозділі?

Завдання підвищеної складності. Яка кількість відпрацьованих днів найчастіше була зафіксована у кожному підрозділі?

Приклад форми для 1-го варіанту показаний на рис. 10.4.

Підрозділ Y				Підрозділ B			
Співробітники	Дні	Оплата	Податок	Співробітники	Дні	Оплата	Податок
1	4	160	32	1	30	1200	240
2	4	160	32	2	19	760	152
3	30	1200	240	3	14	560	112
4	7	280	56	4	2	80	16
5	29	1160	232	5	3	120	24
6	18	720	144	6	4	160	32
7	17	680	136	7	14	560	112
8	20	800	160	8	27	1080	216
9	14	560	112	9	25	1000	200
10	12	480	96	10	27	1080	216
11	29	1160	232	11	9	360	72
12	22	880	176	12	19	760	152
13	25	1000	200	13	10	400	80
14	19	760	152	14	14	560	112
15	14	560	112	15	19	760	152
				16	6	240	48
				17	7	280	56
				18	13	520	104
				19	18	720	144
				20	7	280	56

OK

В 1 підрозділі більше 15 днів працювало: 9
В 2 підрозділі більше 15 днів працювало: 8

Рис.10.4 – Зразок форми для завдання 1

10.3. Завдання 2. Пошук у двовимірних масивах

Простий пошук. Утворити і вивести двовимірний масив $A[4, 4]$. Кожен елемент якого визначається виразом: $a_{kn} = nf_{i+1}(k) + \sin(k)f_{i+2}(n)$, де i – номер варіанта, k, n індекси елементів від 1 до 4. Провести пошук у відповідності до завдання вашого варіанту.

Варіанти завдань:

1. Обчислити добуток від'ємних елементів масиву, та визначити індекси його мінімального елемента.
2. Знайти кількість додатних елементів масиву та їхній добуток.
3. Знайти скільки елементів масиву задовольняють нерівність $1 < a_{kn} < 7$.
4. Знайти добуток значень елементів, для яких виконується нерівність $a_{kn} < -1$ або $a_{kn} > -1$.
5. Знайти всі елементи, які більше ніж 2 і обчислити суму квадратів цих елементів.

6. Вивести добуток квадратів тих елементів масиву, які задовольняють умову $|a_{kn}| < 3$.
 7. Обчислити суму додатних та суму індексів від'ємних елементів.
 8. Обчислити добуток від'ємних елементів. Визначити індекси максимального елемента.
 9. Обчислити суму та добуток діагональних елементів масиву, вивести індекси максимального елемента у діагоналі.
 10. Обчислити суму елементів масиву, для яких виконується умова $2 < a_{kn} < 10$.
 11. Визначити індекси максимального елемента масиву. Визначити суму елементів над головною діагоналлю.
 12. Обчислити добуток елементів перших двох рядків.
 13. Обчислити суму елементів масиву над головною діагоналлю. Визначити індекси мінімального елемента.
 14. Знайти найбільший по модулю від'ємний елемент. Обчислити суму всіх від'ємних елементів.
 15. Вивести добуток елементів, які задовольняють нерівності $a_{kn} < -5$ або $a_{kn} > 3$. Визначити індекси мінімального елемента.
 16. Знайти індекси мінімального додатного і максимального від'ємного елементів масиву.
 17. Знайти суму елементів під головною діагоналлю. Визначити максимальне з цих значень.
 18. Визначити кількість від'ємних і суму додатних елементів.
 19. Визначити індекси максимального та мінімального елементів масиву та суму елементів головної діагоналі.
 20. Вивести їхні індекси мінімального і максимального елементів масиву, знайти їх добуток.
- Зразок форми 1-го варіанту показано на рис. 10.5.

OK			
11,08010	-1,73989	4,22720	-7,81401
17,76300	-8,23134	7,71697	-11,67330
25,24540	-13,85890	11,34080	-16,25160
33,03730	-19,15210	15,01650	-21,10830
Індекси мінімального елемента масиву: 4 4			
Добуток від'ємних елементів масиву: 1,18947E8			

Рис.10.5 – Зразок форми 1-го варіанту для завдання 2

10.4. Завдання 3. Пошук за складним ключем у двовимірних масивах

У шістьох населених пунктах проводились вибори депутатів до місцевої ради. Всього подало заявки п'ять кандидатів на одне вакантне місце.

Кількість голосів, набраних кандидатами у кожному пункті, визначається формулою $random(10i + 100)$, де i - номер варіанта.

Сформувати двовимірний масив із даними про результати голосування та визначити додаткові “статистичні” дані у відповідності до завдання за варіантом.

Варіанти завдань:

1. Скільки проголосувало за кожного кандидата?
2. В яких населених пунктах проголосувало більше ніж 200 виборців?
3. Хто з кандидатів набрав максимальну, а хто - мінімальну кількість голосів у кожному населеному пункті?
4. Скільки виборців проголосувало за першого та п'ятого кандидатів по всім населеним пунктам сумарно?
5. В яких населених пунктах перемогли перший та другий кандидати?
6. Яка явка на вибори у кожному пункті?
7. Хто переміг у другому населеному пункті?
8. В яких населених пунктах явка була більшою ніж 100?
9. Визначити двох кандидатів із найменшою кількістю голосів?
10. В яких населених пунктах переміг перший кандидат?
11. Хто з кандидатів набрав найбільше голосів у 1 і 3 населених пунктах?
12. В якому населеному пункті перший кандидат набрав мінімальну кількість голосів, а в якому — максимальну?
13. В якому населеному пункті проголосувало найбільше людей?

14. У кого із перших трьох кандидатів найвищий рейтинг?
15. Хто набрав максимальну, а хто - мінімальну кількість голосів у першому населеному пункті?
16. Які населені пункти не довіряють кандидатам і всього проголосувало менше 200 громадян?
17. У кого з кандидатів рейтинг більший від деякого заданого числа n ?
18. Який розрив між кандидатом, що набрав найбільше голосів і кандидатом, що набрав найменше голосів в кожному населеному пункті?
19. Хто з кандидатів має максимальний рейтинг, у кожному населеному пункті?
20. Хто переміг на виборах?

Загальний вигляд інтерфейсу додатку у формі показано на рис. 10.6.

The screenshot shows a window titled "Голосування" (Voting) with a sub-window titled "Результати" (Results). Inside, there is a table with the following data:

	Кандидат 1	Кандидат 2	Кандидат 3	Кандидат 4	Кандидат 5
Пункт 1	3	29	1	19	19
Пункт 2	5	7	59	33	59
Пункт 3	20	17	44	5	3
Пункт 4	48	30	2	22	7
Пункт 5	16	44	34	43	54
Пункт 6	24	13	42	43	37
Підсумкові	92	127	140	122	142

Рис.10.6 – Загальний вигляд інтерфейсу додатку для завдання 3

10.5. Зміст звіту

1. Код створених програм (перенести за допомогою Print Screen або скопіювавши). До коду бажано додавати коментарі, що пояснюють основні моменти і особливості виконання програми.
2. Фото із результатами виконання як в консолі, так і у формі.
3. Загальні висновки по роботі.

10.6. Контрольні запитання

1. Як оголосити двовимірний масив? З якого числа починається індексація елементів масивів в C++?
2. У чому принципова різниця між одновимірним та багатовимірними масивами?
3. Як ініціалізувати багатовимірний масив початковими значеннями?
4. Який елемент використовується для візуального подання масиву на формі?
5. Який індекс відповідає за нумерацію рядків у двовимірному масиві? А для елемента StringGrid?
6. Як можна графічно відобразити дані багатовимірних масивів? До якої розмірності це має сенс?
7. Чи може користувач безпосередньо за допомогою мишки редагувати вміст комірок елемента StringGrid?

СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Bjarne Stroustrup. The C++ Programming Language (4th Edition) Addison-Wesley, May 2013.- 1368p. ISBN 978-0321563842.
2. Керниган Б., Ритчи Д. Язык программирования С 2-е издание. : Пер. с англ. - М. : Вильямс, 2009. - 304 с.
3. Шпак З.Я. Програмування мовою С.–Львів: Оріяна-Нова, 2006. – 432 с.
4. Schildt Herbert. C++: The Complete Reference 4th edition.-McGraw-Hill Education, 2003. -1056 p.
5. Дейтел, Харви М. Как программировать на C++.- М.: Бином-Пресс, 2008. – 1454 с.
6. Вінник В.Ю. Алгоритмічні мови та основи програмування. Мова С. – Житомир: ЖДТК, 2007. – 328 с.
7. Архангельский А.Я. Программирование в C++ Builder 6. – М: БИНОМ, 2003. – 1152 с.
8. Уроки програмування на C++ / aCode. – Режим доступу: <https://acode.com.ua/uroki-po-cpp/>