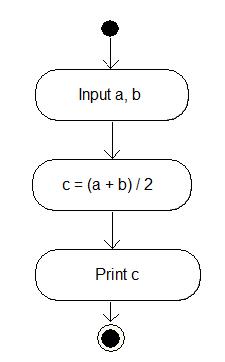
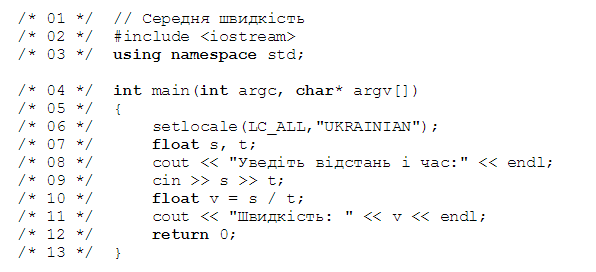
**Лабораторна робота №1**

# Основні концепції програмування. Розробка та графічне подання алгоритмів





Рядок 01 містить *коментар* – деякий пояснювальний текст всередині тексту програми; коментар призначений для того, щоб допомогти людині зрозуміти програмний код. Коментарі – це просто текст, який ігнорується компілятором.

Рядок 02 – це так звана *директива препроцесору*. Препроцесор – це підпрограма, обробляє початковий текст відповідно до її директив і створює так звану *одиницю трансляції* (translation unit). Виконуючи директиву include, препроцесор вставляє текст вказаного *заголовного файлу* в одиницю трансляції. Стандартний заголовний файл iostream містить, зокрема, оголошення стандартного потоку введення cin (клавіатура), стандартного потоку виведення cout(консольне вікно), і так званого маніпулятора endl (кінець рядка).

У рядку 03 здійснюється підключення так званого простору імен. У даному випадку підключається простір імен std. Таке підключення забезпечує можливість використовувати в тексті імена cin, cout і endl безпосередньо, а не std::cin, std::cout і std::endl.

*Примітка*: підключення всіх імен певного простору може призвести до конфліктів імен і тому є прийнятним тільки у невеличких навчальних програмах.

Рядок 04 містить *заголовок функції main()*. Виконання будь-якої програми починається з першої інструкції функції з ім'ям main(). Дужки охоплюють список аргументів. Цілий аргумент argc представляє кількість параметрів командного рядка. Аргумент argc – це вказівник на масив параметрів командного рядка. Якщо ми не бажаємо обробляти аргументи командного рядка, функцію main() можна визначити без аргументів:

**int** main()

{

}

Фігурні дужки в рядках 05 і 13 охоплюють програмний блок функції main(). Програмний блок містить інструкції, які будуть виконані.

У рядку 06 ми викликаємо стандартну функцію setlocale(LC\_ALL,"UKRAINIAN"), яка забезпечує використання відповідної кодової таблиці під час консольного виведення.

*Примітка*: навіть, якщо правильно вказати локалізацію, можуть виникнути проблеми з відображенням української літери і. В усіх прикладах замість української літери доводиться вживати відповідну літеру латинської абетки.

Твердження в рядку 07 – це визначення двох змінних типу **float**. Цей тип використовують для представлення дійсних даних. Змінна s зберігатиме відстань, а t– час.

У рядку 08 послідовність символів у подвійних лапках "Уведiть вiдстань i час:" записується в стандартний вихідний потік виведення cout. Під час виконанні оператора на екран виводиться текст рядка (без лапок). Занесення в потік виведення маніпулятора endl забезпечує перехід курсору на новий рядок.

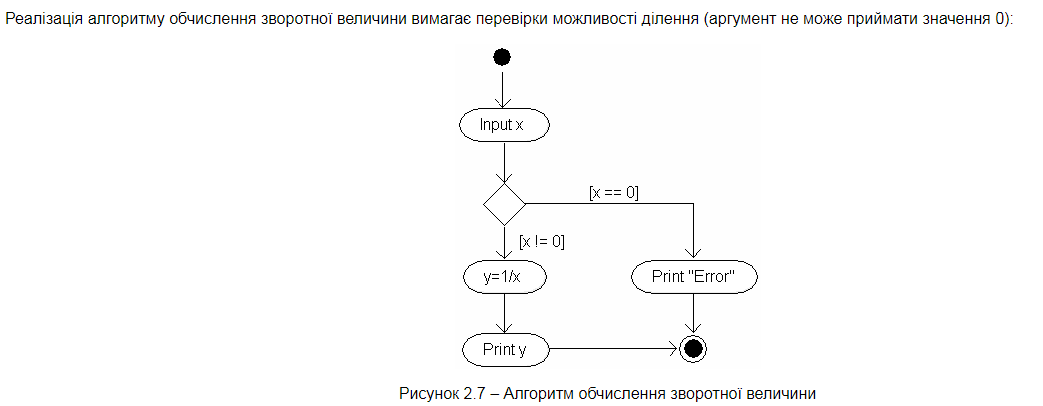
У рядку 09 представлена операція читання значень s і t зі стандартного потоку введення cin (у нашому випадку – з клавіатури). У рядку 10 визначення змінної v об'єднане з обчисленням виразу s / t. У рядку 11 на екран виводиться константа "Швидкiсть: " та значення змінної v.

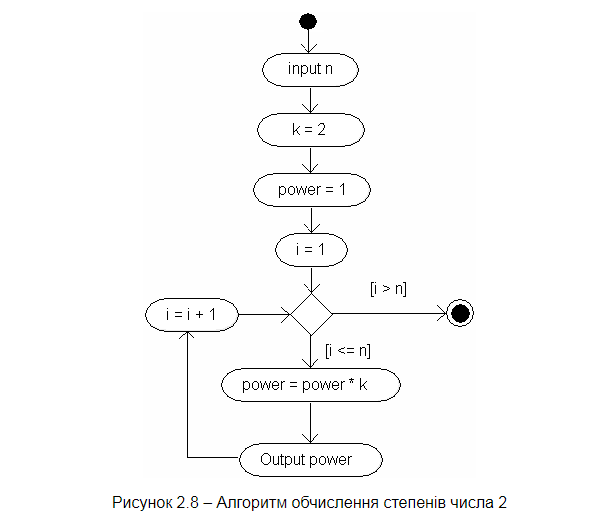
Значення, яке повертає функція main() у рядку 12, може бути використане операційною системою. Значення 0 свідчить про нормальне завершення.

*Примітка*. Якщо завантажити програму на виконання без зневадження (Ctrl-F5), після її завершення у консольному вікні з'явиться додатковий рядок "Press any key to continue . . .". Закінчити роботу можна, натиснувши будь-яку клавішу. Якщо рядок не з'являється, наприклад, через запуск зі зневадженням (F5), до програми можна додати виклик функції, яка очікує на натиснення будь-якої клавіші перед виходом з функції:

system("pause"); // З'явиться "Press any key to continue . . ."

**return** 0;





Наступна програма передбачає введення основи степеня і цілого показника степеня, обчислення і виведення степеня (циклічний алгоритм, аналогічний наведеному на рис. 2.8):

/\* 01 \*/ // Цілий степінь

/\* 02 \*/ #include <iostream>

/\* 03 \*/ **using namespace** std;

/\* 04 \*/ **int** main()

/\* 05 \*/ {

/\* 06 \*/ setlocale(LC\_ALL,"UKRAINIAN");

/\* 07 \*/ **float** x;

/\* 08 \*/ **int** n;

/\* 09 \*/ cout << "Уведiть основу степеня i показник степеня:" << endl;

/\* 10 \*/ cin >> x >> n;

/\* 11 \*/ **float** power = 1;

/\* 12 \*/ **for** (**int** i = 1; i <= n; i++)

/\* 13 \*/ power \*= x;

/\* 14 \*/ cout << "Степiнь: " << power << endl;

/\* 15 \*/ **return** 0;

/\* 16 \*/ }

На відміну від основи степеня (x), показник степеня n визначається в рядку 08 як ціла змінна.

У рядку 11 визначається змінна power типу **float** і їй присвоюється значення 1. Це значення використовується як перше наближення результату.

Рядок 12 містить оператор циклу **for** – циклу з параметром. Параметр і визначається всередині циклу і зміняються від 1 (**іnt** і = 1) з кроком 1 (і++) поки виконується умова і <= n (менше або дорівнює). На кожному кроці циклу виконується інструкція, що міститься безпосередньо після інструкції **for** (...).

Рядок 13 є "тілом циклу". У ньому n разів здійснюється домноження значення змінної power на значення x (в комірку, що містить спочатку 1, n разів записується значення, рівне попередньому, помноженому на x).

**4 Вправи для контролю**

**Завдання 1**

Розробити алгоритм та написати програму, в якій здійснюється читання значення певної довжини в дюймах і обчислюється й виводиться значення цієї довжини в міліметрах (1 дюйм = 25,4 мм).

**Завдання 2**

Розробити алгоритм та написати програму, яка зчитує вісім значень і повертає середнє арифметичне.

**Завдання 3**

Розробити алгоритм та написати програму, яка зчитує значення змінної *n* цілого типу й обчислює *n*!

**6 Контрольні запитання**

1. Що таке позиційна система числення?

Система числення – це сукупність правил і знаків, за допомогою яких можна представити (закодувати) будь-яке число. Найбільш поширені позиційні системи числення – системи, в яких одна і та ж цифра в запису числа має різні значення залежно від позиції, в якій вона розташована.

1. Що таке основа системи числення?

У позиційній системі числення присутнє поняття основи системи числення. Будь-яке число може бути подане як сума ступенів основи, помножених на значення відповідних цифр. Значення основи дорівнює кількості цифр, що використовуються для запису числа. За основу системи числення можна прийняти будь-яке число, не менше, ніж 2. Найменування системи числення відповідає її основі (десяткова, двійкова, вісімкова, шістнадцяткова і т. д.). У повсякденній практиці використовують десяткову система числення.

1. Які є недоліки й переваги двійкової системи числення?

В обчислювальній техніці традиційно прийнята *двійкова система числення*, заснована на представленні всієї інформації у вигляді логічних послідовностей нулів і одиниць. Ця система має дві основні переваги:

* для представлення одиниці використовується електронний сигнал (рівень напруги), відмінний від нуля; оскільки амплітуда сигналу не має значення, це істотно підвищує надійність зберігання і передачі даних;
* правила виконання арифметичних операцій дуже прості й легко можуть бути реалізовані апаратно.

До недоліків двійкової системи можна віднести громіздкість запису чисел і недостатню наочність. У тих випадках, коли для роботи важливо саме двійкове подання числа, використовують системи числення з основою – ступенем двох.

1. Для чого використовують шістнадцяткову систему числення?

Раніше була поширена вісімкова система, в цей час найбільш часто використовується шістнадцяткова система числення. Для переведення із двійкової системи числення в шістнадцяткову кожні чотири двійкових розряди , починаючи справа , замінюються однією шістнадцятковою цифрою. В якості шістнадцяткових цифр прийнято використовувати десять десяткових цифр і букви A (10), B (11), C (12), D (13), E (14) і F (15). Зворотній переклад аналогічний – кожна шістнадцяткова цифра замінюється чотирма двійковими.

1. Визначте поняття програмного забезпечення.

Програмне забезпечення (ПЗ, software) – це сукупність програм і програмних документів, необхідних для експлуатації цих програм.

1. Як можна класифікувати програмне забезпечення?

Загальноприйнятою є така класифікація програмного забезпечення:

* системне ПЗ – програми, що забезпечують управління компонентами комп'ютерної системи; до системного ПЗ можна віднести:
  + операційні системи (operating systems);
  + системні утиліти (utilities);
  + СУБД (системи управління базами даних, database management systems)
  + драйвери пристроїв (device drivers);
* прикладне ПЗ – програми і пакети програм, призначені для розв'язання задач користувачів у конкретних предметних галузях; до прикладного програмного забезпечення відносять текстові процесори, графічні редактори, поліграфічні системи, програми для виконання наукових і технічних розрахунків, ігри і т. д.;
* інструментальні засоби – програмне забезпечення, призначене для створення іншого програмного забезпечення; до інструментальних засобів відносять компілятори та інші утиліти для збирання й налагодження, а також інтегровані середовища розробки (IDE, Integrated Development Environment) і CASE-системи (Computer-Aided Software Engineering).

1. Що таке інструментальні засоби?

Інструментальні засоби – програмне забезпечення, призначене для створення іншого програмного забезпечення; до інструментальних засобів відносять компілятори та інші утиліти для збирання й налагодження, а також інтегровані середовища розробки (IDE, Integrated Development Environment) і CASE-системи (Computer-Aided Software Engineering).

1. Що таке застосунок (додаток)?

Застосунок (додаток, застосування, application) – це синонім прикладної комп'ютерної програми. Однак часто під застосунком розуміють будь-яку програму, яка не є частиною операційної системи.

1. Які функції виконує операційна система?

*Операційна система*– це сукупність програмних засобів, що забезпечують управління всіма ресурсами і процесами обчислювальної системи. Під *ресурсами*розуміють процесор, пам'ять, дисплей, а також периферійні (зовнішні) пристрої.*Процеси* – це окремі програми, що отримали необхідні ресурси і запущені на виконання.

У складі операційної системи виділяють три групи компонентів:

* ядро (планувальник, драйвери пристроїв, підтримка мережі, файлова система);
* системні бібліотеки;
* набір утиліт.

1. Наведіть приклади операційних систем.

Приклади операційних систем – MS-DOS, OS/2, Mac OS, MS Windows різних версій, UNIX, Linux, Solaris, Google Android і багато інших.

1. Що таке файлова система?

Файлова система (file system) – це спосіб і правила організації та зберігання каталогів і файлів на зовнішньому пристрої. Іноді під файлової системою також розуміють сукупність файлів і каталогів на конкретному пристрої.

За структурою і організації доступу до файлів розрізняють *ієрархічні файлові системи* (файли і підкаталоги довільної вкладеності) і *плоскі файлові системи*(набір файлів без підкаталогів). Ієрархічні файлові системи в даний час є найбільш поширеними. Існують спеціальні види файлових систем:

* *кластерні файлові системи*дозволяють розподіляти файли між декількома однотипними фізичними пристроями одного комп'ютера;
* *мережеві файлові системи*забезпечують механізми доступу до файлів одного комп'ютера з інших комп'ютерів мережі;
* *розподілені файлові системи* забезпечують зберігання файлів шляхом їх розподілу між декількома комп'ютерами мережі.

Файлову систему зазвичай розглядають як частину операційної системи. Кожна операційна система надає свій набір файлових систем. Наприклад, файлові системи FAT 16, FAT 32, NTFS пов'язані з Windows, Ext2, Ext3, Ext4 використовують в Linux.

1. Чим текстові файли відрізняються від бінарних?

Незалежно від файлової системи, всі файли можна поділити на текстові та бінарні (двійкові).

*Текстовий файл* (text file) – це комп'ютерний файл, в якому вся інформація представлена ​​у вигляді символів визначеної кодової таблиці. Послідовність символів розділена на рядки. Для відокремлення рядків один від одного використовуються роздільники – один або більше спеціальних керуючих символів. Приклади текстових файлів – прості документи, створені за допомогою блокнота (\*.txt), початкові коди (сирцеві тексти) програм мовами високого рівня (\*.pas, \*.c, \*.cpp, \*.cs, \*.java і т. д.), файли розмітки гіпертексту (\*.htm, \*.html), форматовані документи (\*.rtf) і т. д. Підготовка та редагування текстових файлів, незалежно від їх спеціального формату і призначення, може бути здійснена універсальними текстовими редакторами, наприклад, блокнотом.

*Двійковий файл* (бінарний файл, binary file) – це комп'ютерний файл, в якому числова інформація ​​подана двійковими числами, відповідно до внутрішнього представлення в пам'яті комп'ютера (а не символами, як в текстових файлах). Кожен окремий формат двійкового файлу вимагає спеціального програмного забезпечення. Приклади двійкових файлів – виконувані файли (програми) під усіма операційними системами, растрові зображення (\*.tif, \*.jpeg, \*.png, \*.gif і т. д.), архіви (\*.zip, \*.rar і ​​т. д.), аудіофайли, відеофайли всіх форматів, файли з двійковим кодом (\*.obj, \*.class і т. д.), а також численні спеціальні формати пакетів прикладних програм.

1. Наведіть приклади текстових і бінарних файлів.

Приклади текстових файлів – прості документи, створені за допомогою блокнота (\*.txt), початкові коди (сирцеві тексти) програм мовами високого рівня (\*.pas, \*.c, \*.cpp, \*.cs, \*.java і т. д.), файли розмітки гіпертексту (\*.htm, \*.html), форматовані документи (\*.rtf) і т. д.

Приклади двійкових файлів – виконувані файли (програми) під усіма операційними системами, растрові зображення (\*.tif, \*.jpeg, \*.png, \*.gif і т. д.), архіви (\*.zip, \*.rar і ​​т. д.), аудіофайли, відеофайли всіх форматів, файли з двійковим кодом (\*.obj, \*.class і т. д.), а також численні спеціальні формати пакетів прикладних програм.

1. Чим визначається "рівень" мови програмування?

*Мова програмування* – це спеціальна нотація, за допомогою якої можуть бути записані інструкції, що забезпечують керування роботою комп'ютера. Мови програмування поділяються на низькорівневі та високорівневі.

* *низькорівневі мови* призначені для конкретного комп'ютера і віддзеркалюють його машинні коди; крім мови машинних команд, до низькорівневих мов також належить мова асемблера;
* *високорівневі мови*не залежать від машинного коду конкретного комп'ютера і дозволяють працювати з абстрактними даними.

Мови високого рівня поділяються на основні групи:

* *процедурні* (FORTRAN, ALGOL-60, BASIC, ALGOL-68, PASCAL, C, Modula-2 тощо);
* *об'єктно-орієнтовані* (Simula-67, Smalltalk-80, C++, ADA, Object Pascal, OBERON, Java, C# тощо).

Існують також мови функціонального (логічного), декларативного та інших форм програмування.

1. Як визначити поняття "комп'ютерна програма"?
2. Які існують етапи розробки програми?

Типовими етапами розробки програми є такі:

* *початковий програмний код* (сирцевий текст, source code) програми готується за допомогою текстового редактора;
* початковий код перетворюється в набір *двійкових інструкцій* (binary code); це може бути машинний код, як у C++, або проміжний двійковий код, як у Java;
* скомпільований код збирається з окремих частин (*компонування*, linking) і виконується.

1. Що таке зневадження?

У випадку виникнення помилок на різних етапах розробки, послідовність кроків повторюється. Виконання програми з метою виявлення помилок і перевірки відповідності алгоритму має назву *зневадження* (debug, отладка).

1. Що таке консольний застосунок і чим він відрізняється від інших видів застосунків?

Прикладні програми (застосунки, applications, приложения), які створюють у сучасних середовищах програмування, з точки зору взаємодії з користувачем можна розділити на дві основні групи:

* консольні застосунки (console applications);
* застосунки графічного інтерфейсу користувача (graphical user interface applications, GUI applications).

*Консольні застосунки* виконуються в спеціальному консольному вікні, або в повноекранному режимі. Для введення даних використовують клавіатуру (стандартний пристрій введення). Дані вводяться або як аргументи командного рядку (після імені програми), або під час виконання програми.

Для створення консольних застосунків традиційно застосовують концепцію (парадигму) *програмування, керованого даними*. Загальна структура програми включає введення вихідних даних, обчислення та виведення результатів. Під час обчислення, залежно від вихідних даних та проміжних результатів, може здійснюватися розгалуження, виконання циклічних операцій, виклик підпрограм тощо.

1. У чому переваги і недоліки інтерпретаторів і компіляторів?

*Інтерпретатор* (interpreter) транслює програму рядок за рядком і одразу ж виконує інструкції, зазначені в цих рядках; як приклади можна навести інтерпретатори BASIC, JavaScript; інтерпретатори забезпечують гнучкість і створюють ефект миттєвого виконання, але необхідність багаторазової трансляції раніше інтерпретованих рядків під час виконання істотно знижує ефективність програми;

*Компілятор* (compiler) транслює весь код у набір команд, які може виконати процесор (віртуальна машина). Програми, написані мовами Pascal, C++, C# та багатьма іншими, завжди оброблюються компіляторами.

1. Що таке алгоритм?

*Алгоритм* (algorithm) це детальний опис послідовності дій, спрямованих на розв'язання визначеної задачі. Мета алгоритму повинна бути досягнута за скінчену кількість кроків.

1. Які є способи подання алгоритму?

Є декілька способів опису алгоритмів, наприклад вербальний спосіб або псевдокод. Найбільш наочним є графічний опис. Є дві загальноприйняті форми графічного зображення алгоритмів – традиційна блок-схема та діаграма діяльності. Остання форма є частиною так званої Уніфікованої мови моделювання.

1. Які є різні типи алгоритмів?

Алгоритми зазвичай поділяються на лінійні, алгоритми з розгалуженням і циклічні.

1. Що таке UML?

*Уніфікована мова моделювання* (Unіfіed Modeling Language, UML) – це графічна нотація для визначення, опису, проектування та документування програмних систем, бізнес-систем і інших систем різної природи, в першу чергу пов'язаних з програмним забезпеченням.

1. Що таке діаграма UML?
2. У чому різниця між блок-схемою і діаграмою діяльності?

*Діаграма діяльності* (Actіvіty dіagram) – одна зі стандартних діаграм UML. Цей вид діаграм може бути використаний для зображення алгоритмів. У цьому випадку використовуються наступні елементи діаграми:

* початковий стан (initial state);
* діяльність (activity);
* перехід (transition);
* символ перевірки умови (decision);
* кінцевий стан (end state).

1. У чому різниця між поданням умовних елементів в блок-схемі та діаграмі діяльності?
2. У чому різниця між поданням виведення та розрахунків у нотації діаграм діяльності?
3. Що таке інтегроване середовище розробки?

Мови розробки використовують загальне інтегроване середовище розробки (Integrated Development Environment, IDE). Інтегроване середовище розробки дозволяє програмісту створювати проекти, здійснювати доступ до файлів допомоги, редагувати й компілювати код, виправити помилки компіляції та компонування.

1. Які мови програмування підтримує Visual Studio?

Microsoft Visual Studio .NET – це комплект (suіte) засобів розробки, що включає Visual Basic .NET, Visual C++ .NET, Visual F++ .NET та Visual C# .NET, а також підтримку Web-програмування.

1. Як створити новий проект у Visual Studio?

Ключовим поняттям розробки програми у Visual C++ є *проект* (project). Проект – це набір взаємозалежних вихідних файлів, компіляція і компонування яких дозволяє створити програму або DLL (Dynamic link library). Початкові файли проекту зазвичай зберігаються в окремій теці. Всередині проекту можна створювати декілька *конфігурацій* – узгодженого набору налаштувань, зв'язаних з певною метою діяльності. Кожен проект містить щонайменше дві конфігурації – *конфігурацію налагодження* (debug configuration) і *конфігурацію релізу* (release configuration).

Новий проект може бути створений декількома способами:

* з використанням сторінки Start Page;
* з використанням головного меню FILE | New | Project...;
* з використанням кнопки New Project;
* клавішною комбінацією Ctrl+Shif+N.

З'являється вікно майстра нового проекту (New Project), у якому слід виконати такі дії:

* серед встановлених типів проектів (Installed) обираємо тип проекту Visual C++;
* в середній частині вікна обираємо Empty Project;
* уводимо ім'я проекту в полі Name. Усталене місце розташування для нового проекту визначається автоматично, але в більшості випадків слід змінити диск і теку;
* натискаємо ОК і порожній проект створений.

Якщо замість Empty Project було обрано шаблон Win32 Project, слід здійснити деякі налаштування у майстрі нового проекту (Project Wizard):

* переходимо на закладку Application Settings;
* обираємо опції Console Application і Empty project і натискаємо Finish.

*Примітка*: не слід вживати шаблон Win32 Console Application без додаткових налаштувань, оскільки згенерований код міститиме елементи, специфічні для Microsoft Visual Studio і не буде відповідати стандарту C++.

Щоб створити новий початковий файл, слід виконати такі дії:

* в меню PROJECT обираємо функцію Add New Item... (або Add | New Item... в контекстному меню пункту провідника рішень проекту);
* обираємо тип файлу: C++ File. Рекомендується ввести ім'я файлу в полі Name; в іншому випадку ім'я буде встановлено в Source.cpp;
* натискуємо OK, після чого в панелі редактора відкривається порожній файл.

1. Як зневаджувати програми в Visual Studio?

Засоби налагодження програми використовуються для пошуку і виправлення логічних помилок.

Крім автоматичного виконання програми з метою налагодження передбачений покроковий режим. До тексту програми необхідно додати принаймні одну точку переривання (breakpoint). Це можна зробити за допомогою клавіші F9, коли курсор знаходиться в необхідному рядку. Коли програма виконується в режимі зневадження, точка переривання обумовлює тимчасову зупинку програми. Далі можна зняти програму з використання, або виконувати звичайне чи покрокове виконання.

Після встановлення точок переривання можна завантажити програму в режимі зневадження (DEBUG | Start Debugging або F5). Після зупинки в точці переривання можна здійснювати покрокове виконання або продовжити виконання далі (до кінця або до наступної точки переривання). Для продовження в звичайному (не покроковому) режимі використовують F5. Покрокове виконання програми може відбуватися з заходженням у підпрограми (DEBUG | Step Into, функціональна клавіша F11) і з пропуском (непокроковим виконанням) підпрограм (DEBUG | Step Over, функціональна клавіша F10).

Якщо під час зневадження помістити курсор миші на змінну, її значення можна подивитись у маленькому віконці. Можна також скористатися засобами відображення та редагування в спеціальних підвікнах.

Підвікно Find Symbol Results містить такі закладки:

* закладка Autos відображає інформацію про змінні, які вживані в поточній та попередній інструкціях;
* закладка Locals відображає інформацію про змінні, які є локальними всередині поточної функції;
* закладка Watch 1 дозволяє дивитися і редагувати значення, які зберігаються в змінних.

Якщо у вікні Locals відображається або структуру, поруч з ім'ям змінної з'являється кнопка. Натиснувши на кнопку, ви можете розширити або звузити перегляд змінної. Кнопка відображається знак плюс (+), якщо змінна відображається в скороченому вигляді, і мінус (-), коли вона відображається в розгорнутому вигляді.

Достроково припинити виконання можна за допомогою функції DEBUG | Stop Debugging (Shift-F5). Можна також зупинити програму з одночасним завантаженням і виконанням з початку (DEBUG | Restart або Ctrl-Shif-F5).