Міністерство освіти і науки України
Державний вищий навчальний заклад
“Ужгородський національний університет”
Математичний факультет
Кафедра системного аналізу і теорії оптимізації

ОСНОВИ ПРОГРАМУВАННЯ У C#

Методичні вказівки до лабораторних робіт для студентів І-го курсу
математичного факультету спеціальності "Прикладна математика"

Ужгород – 2014

Брила А.Ю., Антосяк П.П., Глебена М.І., Чупов С.В., Семйон І.В. Основи програмування у C#. Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного факультету спеціальності "Прикладна математика". – Ужгород, 2014. – 60с.

Розглядаються базовий синтаксис та основні програмні конструкції мови C# у рамках структурного підходу до розробки програм. Наведено необхідний теоретичний матеріал та приклади розв’язання деяких задач. Запитання для самоконтролю та завдання для самостійного виконання дозволяють закріпити розглядуваний матеріал.

Рецензенти: канд. фіз.-мат. наук, доц. Мич І.А.,
канд. фіз.-мат. наук, доц. Погоріляк О.О.

Рекомендовано до друку Вченою радою математичного факультету ДВНЗ “Ужгородський національний університет” 20 листопада 2014 року, протокол № 4.

1. ТЕХНОЛОГІЯ .NET FRAMEWORK

Технологія .Net Framework представляє собою середовище, яке дає можливість розробляти і виконувати програми незалежно від платформи (операційної системи). Вона дозволяє розробляти проекти, частини яких можуть бути виконані у різних мовах програмування. Основною мовою програмування цієї технології є мова C#. У цій технології єдиний спосіб опису програмного коду (метадані), одне середовище виконання (Common Language Runtime) і одна базова бібліотека (BCL).

Міжмовна інтеграція досягається за рахунок того, що програма, написана на мові високого рівня (C# чи Basic, чи C++, чи ін.) не одразу компілюється у машинний код, а спочатку відбувається компіляція у команди (псевдокод) єдиної проміжкової мови Intermediate Language (IL).

Незалежність від платформи досягається за рахунок того, що під час виконання програми команди проміжкової мови перетворюються у машинні команди у середовищі Common Language Runtime (CLR) за допомогою відповідних компіляторів (JIT-компіляторів).

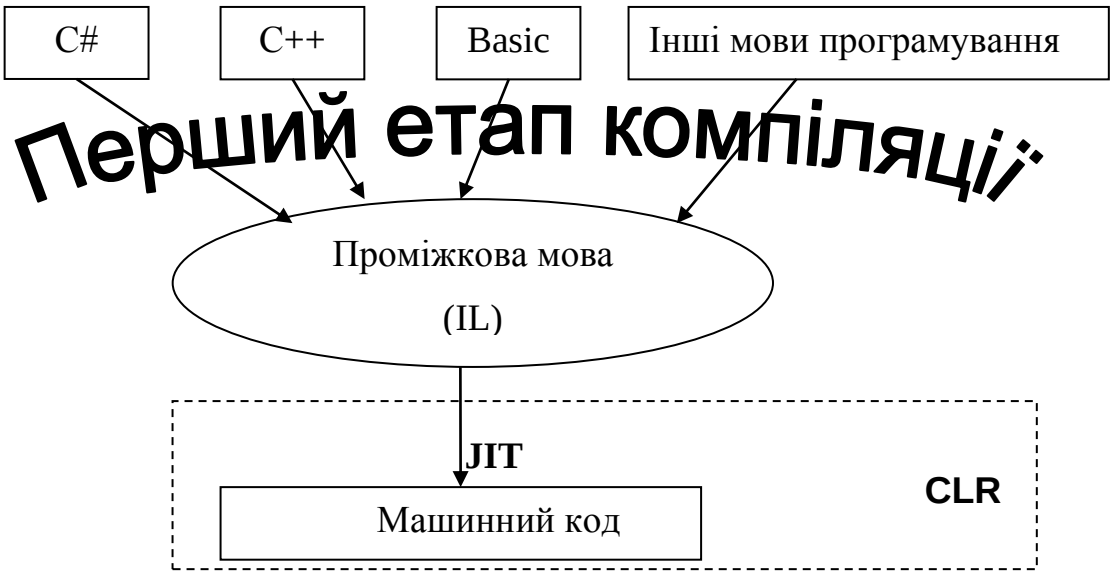


Рисунок 1. Схема виконання програми у .NET

Отже, технологія .Net Framework реалізована в середовищі Common Language Runtime (CLR). CLR можна інтерпретувати, як деякий препроцесор,

який перетворює команди проміжкової мови Intermediate Language (IL) в команди процесора. При цьому використання CLR для розробки програм забезпечує:

- міжмовну інтеграцію;
- автоматичне керування пам’яттю (збирання сміття);
- контроль типів;
- контроль версій програм.

Більшість із цих можливостей можливі завдяки тому, що CLR реалізовано з використанням метаданих. Метадані представляють собою об’єднання програм і даних в одному файлі формату Portable Executable (PE). Програма на проміжковій мові містить не тільки сам програмний код, але і метадані, які надають середовищу виконання інформацію про всі типи, що використовуються, всі сигнатури, всіх елементів цих типів та іншу інформацію, яка в програмах з використанням COM поміщалась в бібліотеки типів і системний реєстр. Таким чином, програма і вся необхідна їй інформація розміщена в одному файлі, що суттєво полегшує установку програми, оскільки виключає реєстрацію в системному реєстрі і пошук відповідних бібліотек.

Як було зазначено, C# є основною мовою програмування технології .Net Framework. Програма може розроблятися і на інших мовах програмування, але при цьому необхідно дотримуватися певних правил, що визначено у середовищі CLR. Так правила описання і використання типів у середовищі CLR визначаються системою віртуальних об’єктів (NGWS Virtual Object System або ж VOS). Використання системи VOS спрощує міжмовну інтеграцію і забезпечує безпечність типів без суттєво зниження роботи програми. В мові використовуються чотири основні складові VOS:

1. Система типів – призначена для уніфікації типів даних, які використовуються різними мовами програмування.
2. Метадані – містять описання і посилання на типи, які визначені в системі.

3. Common Language Specification – специфікація єдиної мови.

4. Virtual Execution System – віртуальне середовище виконання. Це середовище представляє собою реалізацію VOS і відповідає за завантаження і виконання програм, які використовують CLR.

Питання для самоконтролю

1. *Що собою являє технологія .Net Framework?*
2. *За рахунок чого досягається міжмовна інтеграція у .Net Framework?*
3. *За рахунок чого досягається незалежність від платформи у .Net Framework?*
4. *Які основні можливості надає середовище Common Language Runtime?*
5. *Яка мова програмування є основною мовою програмування технології .Net Framework?*

2. МОВА ПРОГРАМУВАННЯ. ПОНЯТТЯ ПРОГРАМИ

З формальної точки зору ***мова програмування (МП)*** **–** це набір вихідних символів *(алфавіт)* разом із системою правил утворення з цих символів формальних конструкцій *(синтаксис)* та системою правил їх тлумачення (інтерпретації) *(семантика),* за допомогою яких описуються алгоритми.

Алфавіт, синтаксис та семантика — це три основні складові частини будь-якої МП. До **алфавіту** МП, як правило, входять літери латинського алфавіту, арабські цифри, знаки арифметичних операцій, розділові знаки, спеціальні символи. Із символів алфавіту будують послідовності, які називають ***словами (лексемами)***. Кожне слово у МП має своє змістове призначення. **Правила синтаксису** пояснюють, як потрібно будувати ті чи інші мовні повідомлення для опису всіх понять мови, здійснення описів та запису вказівок. **Правила семантики** пояснюють, яке призначення має кожен опис та які дії повинна виконати обчислювальна машина під час виконання кожної із вказівок. Вказівки на виконання конкретних дій називають ще ***командами*** або ***операторами*** мови.

Текст алгоритму, записаний засобами конкретної мови програмування, називають ***програмою.***

Усі слова або ж лексеми, поділяють на

- *службові* (зарезервовані або ж ключові) *слова*;
- *стандартні ідентифікатори;*
- *ідентифікатори користувача;*
- *знаки операцій;*
- *розділові знаки;*
- *літерали..*

Службові слова мають наперед визначене призначення і використовуються для формування структури програми, здійснення описів, позначення операцій, формування керуючих конструкцій (вказівок). Наприклад, службовими словами для мови C# є: `abstract`, `do`, `in`, `protected`, `as`, `public`, `try`, `base`, `else`, `interface`, `readonly`, `typeof`, `enum`, `internal`, ref, `break`, `event`, `is`, `return`, `byte`, `explicit`, `lock`, `unchecked`, `case`, `extern`, `sealed`, `unsafe`, `catch`, `namespace`, `finally`, `new`, `sizeof`, `using`, `checked`, `fixed`, `alloc`, `virtual`, `class`, `const`, `for`, `operator`, `volatile`, `continue`, `foreach`, `out`, `struct`, `while`, `goto`, `override`, `switch`, `default`, `if`, `params`, `this`, `delegate`, `implicit`, `private`, `throw`.

Імена (позначення) для програмних об'єктів (типів даних, констант, змінних, функцій і т. п.) формують у вигляді ідентифікаторів. *Ідентифікатор* — це послідовність латинських літер, цифр і знака підкреслення, яка розпочинається з латинської літери. У мові C# максимальна довжина ідентифікатора є необмеженою.

Стандартні ідентифікатори використовуються як імена для стандартних (передбачених авторами мови програмування) типів даних, констант, підпрограм (зокрема, стандартних математичних функцій). Приклад: `int`, `long`, `double`, `char`, `bool` та ін.

Ідентифікатори користувача є іменами тих програмних об'єктів (констант, змінних, функцій тощо), які створює сам користувач. Службові слова та ідентифікатори користувача не повинні збігатися.

Коментар – частина тексту програми, що ігнорується компілятором і служить для деяких поміток у програмі тільки для програміста. Коментарі бувають однорядковими та багаторядковими. Однорядковий коментар починається з символів «//» і закінчується у кінці рядка. Тобто всі символи до кінця рядка вважаються коментарем.

Приклад:

```
// Це однорядковий коментар
```

Багаторядковий коментар починається з символів «/*» і закінчується символами «*/».

Приклад:

```
/* Це
   багаторядковий
   коментар */
```

Знак операції – це один або більше символів, які визначають дію над операндами. Наприклад, у виразі `a += b` знак `+=` є знаком операції, а `a` і `b` – операндами. Символи, які утворюють знак операції, можуть бути як спеціальними, наприклад, `&&`, `|` та `<,` так і буквеними, такими як `as` або `new`.

За кількістю операндів операції поділяють на *унарні*, *бінарні* та *тернарну*. Один і той самий знак операції може інтерпретуватися по різному в залежності від контексту. Всі знаки операцій, за винятком [ ], ( ) та ? :, є окремими лексемами.

Розділові знаки служать для відокремлення однієї лексеми від іншої. Ними є пробіл, табуляція, символ нового рядка, символ «;» та коментарі.

Нижче наведено всі знаки операцій та розділові знаки, які використовуються у мові C#:

```
{ } [ ] ( ) . , : ; + - * / % & | ^ ! ~ =
< > ? ++ -- && || << >> == != <= => += -= *=
/= %= &= |= ^= <<= >>=
```

Літерал – це явно вказане значення декого типу. Розрізняють такі типи літералів:

Тип літерала	Опис	Приклади
цілочисловий	у десятковій системі числення – звичне нам ціле число	125, -89, 108
	у вісімковій системі числення – починається з 0	023, 075, 0416
	у шістнадцятковій системі числення – починається з 0x або 0X	0x4A, 0x6FE2, 0xABC
дійсного типу	у форматі з фіксованою крапкою – у записі числа є крапка, що розділяє цілу і дробову частини	25.69, 2.0, 145.058, 1.2d, 1.2f
	у форматі з плаваючою крапкою – у записі використано символ «e» або «E», що розділяє мантису від порядку	3.5e5, 3e12, 678e2
символьний	заданий у явному вигляді – символ записується у одинарних лапках	‘Z’, ‘a’, ‘E’
	прості ескейп-послідовності – службові символи починаються з символу «\»	\n - перехід на новий рядок, \t - горизонтальна табуляція, \’ - апостроф, і т.д.
	ескейп-послідовності Unicode – символи «\u», за якими вказують код символу з чотирьох цифр у шістнадцятковій системі числення	‘\u0123’, ‘\u3A58’
рядковий	дослівний – починається із символу «@», за яким у подвійних лапках вказується рядок символів (при цьому ескейп-послідовності сприймаються як символи)	@”Я люблю \t C#”
	регулярний – у подвійних лапках вказується рядок символів (ескейп-послідовності обробляються)	”Я люблю \t C#”
логічний	може приймати два значення	true, false
порожньої адреси (нульовий літерал)	використовується у випадку, коли покажчик не містить жодної адреси	null

Питання для самоконтролю

1. *Що таке мова програмування?*
2. *Які три основні складові частини будь-якої мови програмування?*
3. *Що таке програма?*
4. *Назвіть основні різновиди лексем.*
5. *Для чого призначені службові слова?*
6. *Що таке ідентифікатор?*
7. *Для чого використовують у програмі коментарі?*
8. *Що таке літерал?*
9. *Назвіть основні типи літералів.*

3. ДАНІ ТА ТИНИ ДАНИХ.

3.1. Поняття типу даних. Змінні та константи.

Тип даних – це сукупність властивостей певного набору даних, від яких залежать: діапазон значень, якого можуть набувати ці дані, а також сукупність операцій, які можна виконувати над цими даними. Тип даних визначає множину значень до якої належать дані відповідного типу.

З іншого боку **тип даних** – це описання того, яку структуру, розмір мають комірки оперативної пам’яті при зберіганні даних певного типу. Іншими словами тип даних можна розглядати як описання розподілу групи двійкових розрядів.

Елемент даних певного типу – це комірка або комірки оперативної пам’яті, що мають фіксовану адресу, розряди яких розподіляються згідно описання даного типу даних.

Слід зазначити, що тип даних – це абстрактне описання і тому прямо використовуватися він не може. Його використання здійснюється через елементи даних відповідного типу. Наприклад, елемент даних типу `byte` – це комірка оперативної пам’яті з певною адресою, розряди якої декодуються або розуміються згідно описання типу `byte`. Оскільки безпосередньо, через вказівку адрес, працювати з елементами даних незручно, то кожному

елементу даних ставиться у відповідність ідентифікатор (ім’я змінної чи константи), користуючись яким можна здійснювати доступ до елемента даних.

З кожним типом даних зв’язано своє унікальне ім’я (ідентифікатор), яке є синонімом певного описання елементу даних відповідного типу. Наприклад, ідентифікатор `byte` є синонімом опису: 8 послідовних розрядів містить ціле значення без знаку в діапазоні від 0 до 255 (у двійковому вигляді займає 1 байт).

Носіями даних у програмах є *константи, змінні* (значення яких зберігається в оперативній пам'яті) та *файли* (на зовнішніх носіях інформації).

Ідентифікатор вмісту елемента даних певного типу, що може приймати різні значення під час виконання програми називається *змінною*. Якщо ж вміст елемента даних є незмінним (сталим) тоді відповідний ідентифікатор вмісту цього елементу даних називається *константою*.

Отже, у програмах змінна або константа характеризується такими ознаками: *іменем, типом* і *значенням.*

3.2. Стандартні типи даних у C#

В мові підтримується стандартний набір типів даних. Однак кожен тип даних є об’єктом, що не передбачається в стандартному наборі. Для вирішення цього протиріччя в мові розрізнять: ***типи значень (value type)*** і ***типи посилань (reference type)***. Типи значень включать в себе дані простого типу, а також переліковний тип даних і структури. До простих типів відносяться, наприклад, такі типи як `char`, `int`, `float`.

До типів посилань відносяться типи класів, інтерфейсів, масивів та делегатів.

До стандартних «простих» типів у мові C# відносяться наступні:

Тип	Опис	Область значень
`object`	Базовий клас для всіх інших типів	
`string`	Рядковий тип, послідовність символів Unicode	
`sbyte`	8-розрядне ціле число з знаком	-128 до 127

Тип	Опис	Область значень
short	16-розрядное ціле число з знаком	-32768 до 32767
int	32-розрядне ціле число з знаком	-2147483648 до 2147483647
long	64-розрядне ціле число з знаком	-9223372036854775808 до 9223372036854775807
byte	8-розрядне ціле число без знака	0 до 255
ushort	16-розрядне ціле число без знака	0 до 65535
uint	32-розрядне ціле число без знака	0 до 4294967295
ulong	64-розрядне ціле число без знака	0 до 18446744073709551615
float	Число з плаваючою крапкою 4 байти, точність — 7 розрядів	±1,5*10-45 до ±3,4*1033
double	Число з плаваючою крапкою 8 байт, точність —16 розрядів	±5*10-324 до ±1,7*10306
bool	Логічний тип	true або false
char	Тип символу Unicode	U+0000 до U+ffff
decimal	Тип десяткового числа 12 байт, точність — 28 розрядів	(Від -7,9 x 10^{28} до 7,9 x 10^{28}) / ($10^{0–28}$)

Зазначимо, що кожен із наведених типів є скороченням деякого системного типу. Наприклад, тип `Int` є альтернативною назвою структури `System.Int32`. В більшості випадків ці імена повністю рівноправні.

3.3. Опис змінних та констант

Опис змінних

Загальне правило опису:

- *без початкової ініціалізації*

```
<тип змінної> <ідентифікатор змінної>;
```

- *з початковою ініціалізацією*

```
<тип змінної> <ідентифікатор змінної> = <значення>;
```

Приклади:

```
int a;          // Без початкової ініціалізації.
int b=40;       // Змінна b ініціалізовується значенням 40.
double s=3.67;  // Змінна s ініціалізовується значенням 3.67.
```

Опис констант

Загальне правило опису:

```
const <тип константи> <ідентифікатор констати> = <значення>;
```

Приклади:

```
const  int  a  = 10;    // константа цілого типу
const  float  b  = 2.5; // константа дійсного типу
```

3.4. Перетворення типів

В мові розрізняють два види перетворень (приведень) змінних – *явне* та *неявне.*

Неявне перетворення типів використовується в тому випадку, коли дане перетворення є «природним». Тобто, наприклад, перетворюється величина типу `float` в величину типу `double`.

Приклад:

```
float f = 1.23;
double d = f;   // Неявне перетворення
```

Таке перетворення є природним, оскільки обидва типи використовуються для представлення дійсних типів, причому цільовий тип має більший діапазон представлення і більшу точність. При такому перетворення не відбувається втрата інформації. При проведенні неявного перетворення немає необхідності вказувати цільовий тип (тип до якого здійснюється перетворення).

Явне перетворення типів вимагає явного задання цільового типу, до якого здійснюється перетворення. Цільовий тип вказується в дужках перед значенням тип якого перетворюється.

Загальне правило явного перетворення:

```
<змінна> = (<цільовий тип>) <вираз>;
```

Приклад:

```
double d = 2.9;
float f = (float)d;  // Явне перетворення
```

Для цільового типу повинно існувати неявне перетворення до типу змінної, в якій буде збережено результат.

3.5. Переліковний тип

Переліковний тип це тип, який задається повним переліком своїх значень. Значеннями переліковного типу можуть бути константи типів `byte`, `sbyte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`. Якщо при описані переліковного типу в явному вигляді не вказано тип його констант, то вважається, що константи типу `int`. Всі константи повинні бути одного типу,

який називається *базовим типом*. Тип `char` не може використовуватися в якості базового типу. В переліковному типі можна задавати довільну кількість іменованих констант.

Загальне правило опису:

```
<атрибути> <модифікатори> enum <ім’я типу> [: <базовий тип>]
{
      <атрибути> <конст.1> [= <знач.1>],
      <атрибути> <конст.2> [= <знач.2>],
      ……
      <атрибути> <конст.N> [= <знач.N>]
}
```

Приклад:

```
enum Week : Byte
{
      Monday,      // =0
      Tuesday,     // =1
      Wednesday,   // =2
      Thursday,    // =3
      Friday,      // =4
      Saturday,    // =5
      Sunday       // =6
}
```

Всі константи переліковного типу ініціалізуються значеннями базового типу. Ця ініціалізація може бути *явною* або *неявною*. За замовчуванням застосовується неявна ініціалізація (дивись приклад вище), при цьому, перша константа у переліку ініціалізується нулем, друга − одиницею, і т.д.

Також константи можуть бути проініціалізовані явно довільними значеннями із базового типу або ж значенням іншої константи із цього ж переліку. При цьому ініціалізація констант не обов’язково повинна здійснюватися послідовними значеннями, не обов’язково ініціалізувати явно всі константи та не обов’язково ініціалізувати константи унікальними значеннями.

Приклад:

```
enum Week : Byte
{
      Monday    =2,
      Tuesday   =35,
```

```
        Wednesday =2,
        Thursday,           // =3
        Friday,             // =4
        Saturday,           // =5
        Sunday    =1,
        Sabbath   =Saturday // =5
}
```

Ініціалізація змінної переліковного типу значенням здійснюється через ім’я типу та оператор "." (крапка).

Приклад:

```
Week d = Week.Friday;
```

Також між змінними переліковного типу та змінними відповідного базового типу існує явне перетворення типів.

Приклади:

```
Week d = Week.Monday;
byte b = (byte)d;           // b = 2
byte i = 1;
Week s = (Week)i;           // s = Week.Sunday
```

Питання для самоконтролю

1. *Що розуміють під терміном "величина" у програмуванні?*
2. *Дайте означення константи та змінної.*
3. *Дайте означення типу даних.*
4. *На які два різновиди поділяють усі типи у мові C#?*
5. *Які типи належать до типів посилань?*
6. *Назвіть та опишіть прості типи мови C#.*
7. *Яке загальне правило опису змінних?*
8. *Яке загальне правило опису констант?*
9. *Яке загальне правило опису переліковного типу?*
10. *У яких випадках використовується неявне перетворення типів?*
11. *Який загальний синтаксис явного перетворення типів?*

4. ВИРАЗИ ТА ОПЕРАЦІЇ

Виразом називають композицію операндів та операцій над ними. Кожен вираз визначає результат певного типу. В залежності від типу результату вирази, найчастіше, поділяють на *арифметичні, логічні та символьні*.

4.1. Арифметичні вирази

Арифметичним виразом називають вираз, в результаті обчислення якого одержуємо числове значення.

4.1.1. Бінарні та унарні операції

У арифметичних виразах використовуються наступні *бінарні* арифметичні операції:

Операція	Позначення	Приклад
+	додавання	`z=x+y`
–	віднімання	`z=x-y`
*	множення	`z=x*y`
/	ділення	`z=x/y`
%	остача від ділення	`z=x%y`

Окрім бінарних операцій в арифметичному виразі можуть бути присутні також *унарні* операції «+», «–» та операції інкременту «++» і декременту «– –». Унарний мінус використовується для зміни знаку. Операції інкременту і декременту використовують для збільшення та зменшення значення змінної на одиницю. Ці операції можуть вживатися у різних формі.

Операція	Аналог
Постфіксна форма	
`i++`	`i=i+1`
`i--`	`i=i-1`
Префіксна форма	
`++i`	`i=i+1`
`--i`	`i=i-1`

Якщо унарну операцію вжито у префіксній формі у виразі, то вона виконується до використання значення змінної у виразі. Якщо ж унарну операцію вжито у постфіксній формі, то операція виконується після використання значення змінної у виразі.

Операція	Аналог з бінарними операціями

`int i = 5;`
 	
`int j = ++i;  //  j=6  i=6`
	`int i = 5;`
	`i=i+1;`
	`int j = i;  //  j=6  i=6`

`int i = 5;`
 	
`int j = i++;  //  j=5  i=6`
	`int i = 5;`
	`int j = i;`
	`i=i+1;  //  j=5  i=6`

`int i = 5;`
 	
`int j = --i;  //  j=4  i=4`
	`int i = 5;`
	`i=i-1;`
	`int j = i;  //  j=4  i=4`

`int i = 5;`
 	
`int j = i--;  //  j=5  i=4`
	`int i = 5;`
	`int j = i;`
	`i=i-1;  //  j=5  i=4`

4.1.2. Побітові операції

У мові C# є можливість здійснювати побітові операції над розрядами аргументів.

Операція	Позначення	Приклад

&	побітове «і»	`int x=10; //x=1010(2)`
		`int y=7;  //y=0111(2)`
		`z=x&y  // z=2=0010(2)`

|	побітове «або»	`int x=10; //x=1010(2)`
		`int y=7;  //y=0111(2)`
		`z=x|y  //z=15=1111(2)`

^	Побітове «виключаюче або»	`int x=10; //x=1010(2)`
		`int y=7;  //y=0111(2)`
		`z=x^y  //z=13=1101(2)`

4.1.3. Операції зсуву

Операції зсуву застосовуються до цілочислових операндів. Вони зсувають двійкове представлення першого операнда вліво або вправо на кількість двійкових розрядів, яка задається другим операндом.

При зсуві вліво (<<) вільні біти обнуляються. При зсуві вправо (>>) вільні біти заповнюються нулями, якщо перший операнд беззнакового типу (логічний зсув), і знаковим розрядом − в протилежному випадку (арифметичний зсув).

Операція	Позначення	Приклад

>>	зсув розрядів вправо
	(змінна) =(змінна)>>(кільк. розрядів)
		`int i=4;//i=100(2)`
		`i=i>>1; //i=2=10(2)`

<<	зсув розрядів вліво
	(змінна) =(змінна)<<(кільк. розрядів)
		`int i=4;//i=100(2)`
		`i=i<<2; //i=16=10000(2)`

4.1.4. Операції присвоєння

Якщо при виконанні бінарної операції результат зберігається у змінній, що є першим аргументом, то можна використати так звані операції присвоювання.

Операція	Приклад	Аналог з бінарними операціями
+=	`int i = 5;` `i += 3;  //  i=8`	`int i = 5;` `i=i+3;  //  i=8`
-=	`int i = 5;` `i -= 3;  //  i=2`	`int i = 5;` `i=i-3;  //  i=2`
*=	`int i = 5;` `i *= 3;  //  i=15`	`int i = 5;` `i=i*3;  //  i=15`
/=	`int i = 6;` `i /= 3;  //  i=2`	`int i = 6;` `i=i/3;  //  i=2`
>>=	`int i = 5;` `i >>= 1;  //  i=2`	`int i = 5;` `i=i>>1;  //  i=2`
<<=	`int i = 5;` `i <<= 1;  //  i=10`	`int i = 5;` `i=i<<1;  //  i=10`

&=	`int i = 5;`
	`int j = 7;`
	`i &= j;  //  i=5`
		`int i = 5;`
		`int j = 7;`
		`i=i&j;  //  i=5`

^=	`int i = 5;`
	`int j = 7;`
	`i ^= 1;  //  i=2`
		`int i = 5;`
		`int j = 7;`
		`i=i^j;  //  i=2`

|=	`int i = 5;`
	`int j = 7;`
	`i |= 1;  //  i=7`
		`int i = 5;`
		`int j = 7;`
		`i=i|j;  //  i=7`

4.1.5. Математичні функції

Арифметичні вирази можуть містити різні математичні функції.

Назва	Опис	Результат	Пояснення
Тригонометричні функції			
Sin	Синус	double	Math.Sin(double x)
Cos	Косинус	double	Math.Cos(double x)
Tan	Тангенс	double	Math.Tan(double x)
Обернені тригонометричні функції			
ASin	Арксинус	double	Math.ASin(double x)
ACos	Арккосинус	double	Math.ACos(double x)
ATan	Арктангенс	double	Math.ATan(double x)
ATan2	Арктангенс	double	Math.ATan2(double x, double y) – кут, тангенс якого є результатом ділення y на x
Гіперболічні функції			
Tanh	Тангенс гіперболічний	double	Math.Tanh(double x)
Sinh	Синус гіперболічний	double	Math.Sinh(double x)
Cosh	Косинус гіперболічний	double	Math.Cosh(double x)

Назва	Опис	Результат	Пояснення
Експонента і логарифмічні функції			
Exp	Експонента	double	Math.Exp(x)
Log	Логарифм натуральний	double	Math.Log(x)
Log10	Логарифм десятковий	double	Math.Log10(x)
Модуль(абсолютна величина), корінь квадратний, знак			
Abs	Модуль	Залежить від типу аргументу	Math.Abs(x)
Sqrt	Квадратний корінь	double	Math.Sqrt(x)
Sign	Знак числа	int	Math.Sign(x)
Заокруглення			
Ceiling	Заокруглення до більшого цілого	double	Math.Ceiling(double x)
Floor	Заокруглення до меншого цілого	double	Math.Floor(double x)
Round	Заокруглення	Залежить від типу аргументу	Math.Round(x)
Мінімум, максимум			
Min	Мінімум двох чисел	Залежить від типу аргументу	Math.Min(x,y)
Max	Максимум двох чисел	Залежить від типу аргументу	Math.Max(x,y)
Степінь, остача			
Pow	Піднесення до степеня	double	Math.Pow(x,y)
IEEERemainder	Остача від ділення	double	Math.IEEERemainder(double x, double y)
Добуток двох цілих величин			
BigMul	Добуток	long	Math.BigMul(int x,int y)
Ділення і остача від ділення			
DivRem	Ділення і остача	Залежить від типу аргументу	Math.DivRem(x,y, rem)
Константи			
E	База натурального логарифма	double	Math.E
PI	Значення числа π	double	Math.PI

4.2. Логічні вирази

Логічним виразом або ***умовою*** називається такий вираз, внаслідок обчислення якого одержується логічне значення типу `bool`.

4.2.1. Операції відношення

Логічний вираз може містити операції відношення.

Операція	Позначення	Приклад
==	рівність	x==y
>	більше	x>y
<	менше	x<y
>=	більше або рівно	x>=y
<=	менше або рівно	x<=y
!=	не рівно	x!=y

4.2.2. Логічні операції

У логічному виразі можуть також використовуватися логічні операції.

x	y	x&&y (логічне «і»)	x || y (логічне «або»)	!x (заперечення)
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

4.3. Пріоритет операцій

При обчисленні значення арифметичних та логічних виразів оператори виконуються у певному порядку. Нижче наведено порядок виконання усіх операторів у мові C#.

1. (), [], ., (постфікс)++, (постфікс)--, new, sizeof, typeof , unchecked
2. !, ~, (ім'я типу), +(унарний), -(унарний), ++(префікс), --(префікс)
3. *, /, %
4. +, -
5. <<, >>
6. <, >, <=, >=, is
7. ==, !=
8. &
9. ^
10. |
11. &&
12. ||
13. ?:
14. =, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=

Питання для самоконтролю

1. *Що називають арифметичним виразом?*
2. *Що називають логічним виразом?*
3. *Які операції може містити арифметичний вираз?*
4. *Які операції може містити логічний вираз?*
5. *Назвіть основні математичні функції, які доступні у мові C#.*
6. *Який пріоритет операцій у мові C#.*

5. КОНСОЛЬНІ ПРОГРАМИ. РЕАЛІЗАЦІЯ ЛІНІЙНИХ АЛГОРИТМІВ

Консоллю або консольною програмою називають вікно операційної системи, в якій користувачі взаємодіють з операційною системою шляхом вводу даних з клавіатури та виводу даних в консольне вікно.

Точкою входу в консольну програму на мові C# є метод Main.

Загальний вигляд	Приклад

class <назва класу>
{
static void Main()
{
<Тіло програми>
}
}
	class Hello
	{
	static void Main()
	{
	Console.WriteLine(“Hello!”);
	Console.ReadKey();
	}
	}

5.1. Введення/виведення даних у консольних програмах

Введення та виведення даних у консольних програмах може бути здійснено з використанням класу `Console`.

5.1.1. Виведення даних

Для виведення даних можуть бути використані методи `Console.Write()` та `Console.WriteLine()`. Метод `Console.WriteLine()` відрізняється від метода `Console.Write()` тим, що після виведення інформації курсор консолі переводиться на наступний рядок.

Метод `Console.WriteLine()`, як і `Console.Write()` мають декілька варіантів виклику, що відрізняються списком аргументів. Розглянемо деякі з варіантів виклику.

Виведення рядка тексту

Загальне правило:

```
Console.WriteLine(<рядок тексту>);
```

Приклад	На екрані
`Console.WriteLine(“Hello!”);`	Hello!

Виведення числового значення

Загальний вигляд:

```
Console.WriteLine(<вираз>);
```

Приклад	На екрані
`int i=25;` `Console.WriteLine(i);`	25

Виведення значення з текстовим повідомленням

Загальний вигляд:

```
Console.WriteLine(<текст>+<вираз>);
```

Приклад	На екрані
`int i=25;` `Console.WriteLine(“i=”+i);`	`i=25`

`int j=3;`
`double d=2.8;`
`Console.WriteLine("d="+d+" j="+j);`
	`d=2.8 j=3`

Виведення із використанням підстановочних знаків. При цьому способі виведення спочатку вказується рядок символів, у якому у фігурних дужках фіксується місце розташування аргументів для виведення (вказуємо номер аргументів, які нумеруються починаючи від 0), а потім, послідовно через кому перераховуємо аргументи для виведення.

Приклад	На екрані
`int i=25;` `Console.WriteLine(“i={0}”,i);`	`i=25`

`int j=3;`
`double d=2.8;`
`Console.WriteLine("d={0} j={1}",d,j);`
	`d=2.8 j=3`

Форматований вивід значень. Цей спосіб виведення можливий з використанням підстановочних знаків з параметрами. Розглянемо деякі з них:

d – десятковий формат. Дозволяє задати загальну кількість знаків (якщо кількість символів у числі менша, то зліва число доповнюється нулями).

Приклад	На екрані
`int i=25;` `Console.WriteLine(“i={0:d5}”,i);`	i=00025

f – формат з фіксованою точністю. Дозволяє задати кількість знаків після коми.

Приклад	На екрані
`double d = 2.853;` `Console.WriteLine("d={0:f2} ", d);`	`d=2.85`
`double d = 2.857;` `Console.WriteLine("d={0:f2} ", d);`	`d=2.86`

x – шістнядцятковий формат. Дозволяє виводити значення аргументу як шістнадцяткове число. Якщо параметр “x”, то цифри виводяться у нижньому регістрі, якщо ж параметр “X”, то цифри виводяться у верхньому регістрі.

Приклад	На екрані
`int i = 255;` `Console.WriteLine("i={0:x} ", i);`	`i=ff`
`int i = 255;` `Console.WriteLine("i={0:X} ", i);`	`i=FF`

c – грошовий формат (додає знак грошової одиниці, у залежності від налаштувань операційної системи).

Приклад	На екрані
`int i = 255;` `Console.WriteLine("i={0:c} ", i);`	`i=255.00р.`

e – вивід числа в експоненціальній формі.

Приклад	На екрані
`double d = 362.853;` `Console.WriteLine("d={0:e} ", d);`	`d=3.628530e+002`

5.1.2. Введення даних

Введення даних можна здійснити використовуючи методи `Console.ReadKey()`, `Console.Read()` та `Console.ReadLine()`.

Введення символьних даних

`Console.ReadLine()` дозволяє зчитати рядок символів із стандартного потоку вводу.

Приклад:

```
string s;
s=Console.ReadLine();
```

`Console.Read()` дозволяє зчитати код символу із стандартного потоку вводу.

Приклад	На екрані

`int c;`
`c =Console.Read();`
`Console.WriteLine(c);`
	
	`A`
	`65`

`char c;`
`c =(char)Console.Read();`
`Console.WriteLine(c);`
	
	`A`
	`A`

Введення числових даних

Для введення числових даних можна використати метод `Console.ReadLine()` для введення рядка символів та подальшого переведення рядкового представлення числа у числове значення відповідного типу. Переведення рядкового представлення числа у число можна використовуючи метод `Parse()` відповідного цільового типу

```
<Цільовий тип>.Parse(<Вираз>);
```

або методи класу `Convert`

```
Convert.To<Цільовий тип>(<Вираз>);
```

Приклади:

З використанням методу Parse	З використанням класу Convert

`int c;`
`Console.Write("c=");`
`c=int.`**`Parse`**`(Console.ReadLine());`
	`int c;`
	`Console.Write("c=");`
	`c=`**`Convert.ToInt16`**`(Console.ReadLine());`

`double d;`
`Console.Write("d=");`
`d=double.`**`Parse`**`(Console.ReadLine());`
	`double d;`
	`Console.Write("d=");`
	`d=`**`Convert.ToDouble`**`(Console.ReadLine());`

5.2. Реалізація лінійних алгоритмів

Лінійним називається алгоритм, всі команди якого виконуються послідовно в тому порядку, в якому вони записані. Як правило, реалізація лінійного алгоритму складається із трьох етапів: ввід вихідних (початкових) даних, обчислення значень виразів та вивід результатів.

Приклад. Обчислити площу трикутника, якщо трикутник задано довжинами сторін.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // введення вихідних даних
            Console.Write("a=");
            double a = double.Parse(Console.ReadLine());
            Console.Write("b=");
            double b = double.Parse(Console.ReadLine());
            Console.Write("c=");
            double c = double.Parse(Console.ReadLine());
            // реалізація алгоритму розв'язання задачі
            double p = (a + b + c) / 2;
            double s = Math.Sqrt(p * (p - a) * (p - b) * (p - c));
            // вивід результату
            Console.WriteLine("s={0}", s);//
            Console.ReadKey();//
        }
    }
}
```

Результат роботи програми:

```
a=3
b=4
c=5

s=6
```

Питання для самоконтролю

1. *Що називають консоллю?*
2. *Як організувати вивід символьних значень на екран?*
3. *Як організувати вивід числових значень на екран?*
4. *Як організувати форматований вивід?*
5. *Як організувати ввід символьних даних?*
6. *Як організувати ввід числових даних?*
7. *Який алгоритм називається лінійним?*

Завдання для самостійної роботи

Завдання 1

№ варіанта	Умова завдання
1	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$. Скласти програму для знаходження площі трикутника за формулою Герона.
2	Обчислити площу трикутника, якщо трикутник задано довжинами сторін.
3	Обчислити площу трикутника, якщо трикутник задано довжиною однієї з сторін та висотою, опущеною на неї.
4	Обчислити площу трикутника, якщо трикутник задано двома сторонами та кутом між ними.
5	Обчислити площу та периметр квадрата, якщо задано довжину сторони цього квадрата.
6	Обчислити площу та периметр квадрата, якщо задано довжину діагоналі цього квадрата.
7	Обчислити площу та периметр прямокутника, довжини сторін якого задаються.
8	Обчислити площу та периметр ромба, якщо задано довжину сторін та один з кутів.
9	Обчислити площу та периметр рівнобічної трапеції, для якої задано довжини основ та висоту..
10	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$. Скласти програму для знаходження периметра.
11	Дано два вектори $x, y \in R^3$. Знайти косинус кута між ними.
12	Дано: $x, y \in R^3$. Знайти $z = \langle x, y \rangle x$ де $\langle x, y \rangle$ – скалярний добуток векторів x та y.
13	Дано: $x, y \in R^3$. Знайти $z = 2x + 3y$.
14	Дано два дійсних числа. Знайти суму, добуток, середнє арифметичне та середнє геометричне цих чисел.
15	Дано катети прямокутного трикутника. Знайти його гіпотенузу та площу.
16	Знайти площу рівнобічної трапеції з основами a і b та кутом α при більшій основі.
17	Знайти площу кільця, внутрішній радіус якого дорівнює 20, а зовнішній заданому числу R (R>20).
18	Трикутник задано довжинами сторін. Знайти довжини висот.
19	Трикутник задано довжинами сторін. Знайти довжини медіан.
20	Трикутник задано довжинами сторін. Знайти радіуси вписаного та описаного кіл.

6. РЕАЛІЗАЦІЯ АЛГОРИТМІВ ІЗ РОЗГАЛУЖЕННЯМ

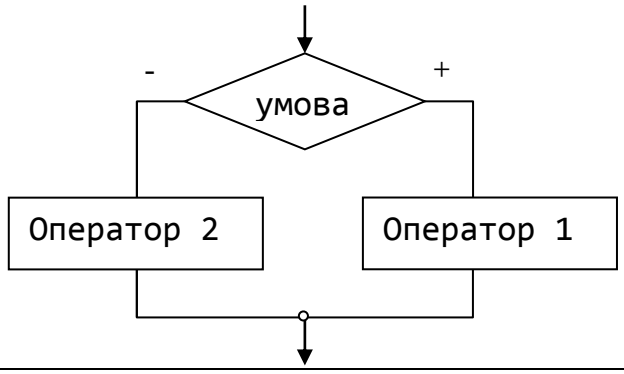
Розгалуженням називається алгоритмічна конструкція, що дозволяє виконавцеві алгоритму вибирати ту чи іншу послідовність дій залежно від певних умов.

6.1. Умовний оператор

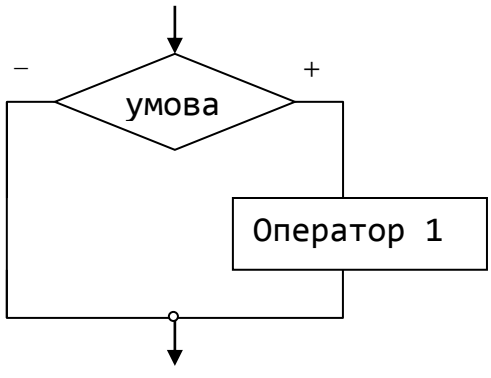
Алгоритмічна конструкція вибору з двох альтернатив, дозволяє виконавцеві алгоритму вибирати один із двох варіантів дій залежно від істинності деякої умови. У мові C# така конструкція реалізується *умовним оператором* (*оператором розгалуження*). Існують дві форми для даного оператора *повна* та *скорочена*.

Повна форма

Програмна структура	**Аналог на мові блок-схем**	**Приклад**

`if (<умова>)`
`  <оператор1>;`
`else`
`  <оператор2>;`
		`if(x>y)`
		`  max=x;`
		`else`
		`  max=y;`

Скорочена форма

Програмна структура	**Аналог на мові блок-схем**	**Приклад**
`if (<умова>)` `  <оператор1>;`		`if(x!=0)` `  z=1/x;`

Приклад. Визначити чи належить значення дійсної змінної x проміжку [0,1] та вивести відповідне повідомлення.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // введення вихідних даних
            Console.Write("x=");
            double x = double.Parse(Console.ReadLine());
            // реалізація алгоритму розв'язання задачі
            if (x >= 0 && x <= 1)
                Console.WriteLine("належить");
            else
                Console.WriteLine("не належить");
            //
            Console.ReadKey();
        }
    }
}
```

Результат роботи програми:

```
x=0,3

належить
```

Гілки деякого розгалуження можуть містити інші розгалуження. У цьому випадку виникає *вкладеність умовних операторів*.

Приклад. Скласти програму для обчислення значення функції

$$f(x)=\begin{cases}0, \text{ якщо } x\le 0,\\ x^2-x, \text{ якщо } 0<x\le 1,\\ x^2-\sin\pi x^2, \text{ якщо } x>1.\end{cases}$$

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // введення вихідних даних
```

```
            Console.Write("x=");
            double x = double.Parse(Console.ReadLine());
            // реалізація алгоритму розв'язання задачі
            double f;
            if (x <= 0)
                f = 0;
            else
                if (x <= 1)
                    f = x * x - x;
                else
                    f = x * x - Math.Sin(Math.PI * x * x);
            // вивід результату
            Console.WriteLine("f={0}", f);
            //
            Console.ReadKey();
        }
    }
}
```

Результат роботи програми:

```
x=0,5

f=-0,25
```

6.2. Складений оператор

Часто виникає потреба у розгалуженнях, гілки яких містять більше ніж один оператор. У цьому випадку застосовують *складений оператор* або *операторні блоки*. *Складений оператор* – це оператор, який об’єднує декілька операторів в одну логічну групу.

Загальне правило запису складеного оператора:

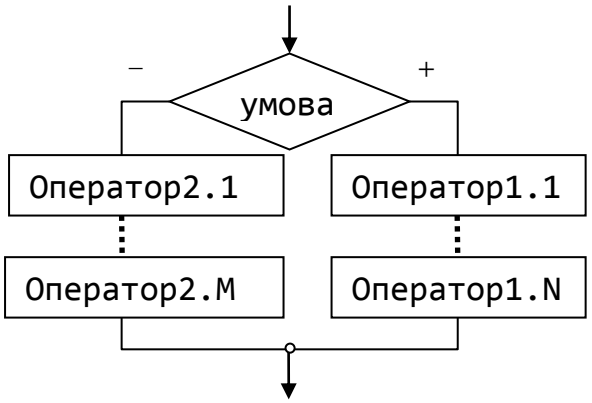
```
{
      <оператор 1>;
      <оператор 2>;
      ……..
      <оператор n>;
}
```

В даній конструкції “{“ – відкриваюча операторна дужка; “}” – закриваюча операторна дужка. Складений оператор визначається як єдиний оператор. Його можна вставляти в довільне місце програми, де дозволено використання одного простого оператора.

Якщо в умовному операторі при виконанні чи невиконанні умови необхідно виконати декілька операторів, то необхідно ці оператори помістити в складений оператор.

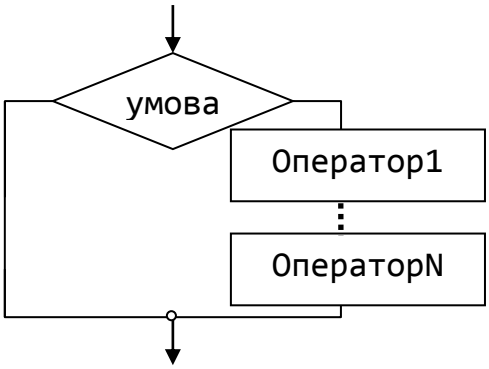
Повна форма

Програмна структура	Аналог на мові блок-схем	Приклад

if (<умова>)
{
<оператор1.1>;
. . . . . .
<оператор1.N>;
}
else
{
<оператор2.1>;
. . . . . .
<оператор2.M>;
}
		if(x>y)
		{
		max=x;
		min=y;
		}
		else
		{
		max=y;
		min=x;
		}

Скорочена форма

Програмна структура	Аналог на мові блок-схем	Приклад

if (<умова>)
{
<оператор1>;
. . . . . .
<операторN>;
}
		if(x>0)
		{
		z=1/x;
		l=y/x;
		}

Приклад. Знайти максимальне та мінімальне із двох дійсних чисел.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {   double b,d;
            Console.Write("b=");
            b = double.Parse(Console.ReadLine());
```

```
                Console.Write("d=");
                d = double.Parse(Console.ReadLine());
                double max, min;
                if (b > d)
                {   max = b;
                    min = d;
                }
                else
                {   max = d;
                    min = b;
                }
                Console.WriteLine("max={0}  min={1}", max, min);
                Console.ReadLine();
            }
        }
    }
```

Результати роботи програми:

```
b=3
d=8

max=8  min=3
```

6.3. Умовний оператор (?:)

Якщо значення виразу може дорівнювати одному із двох значень в залежності від виконання чи невиконання деякої умови, то можна скористатися умовним оператором **(?:)**.

Загальний вигляд умовного оператора (?:)	**Аналог з використанням умовного оператора if**

`<змінна>=<умова>?<значення1>:<значення2>;`	`if (<умова>)`
	`<змінна>=<значення1>;`
	`else`
	`<змінна>=<значення2>;`

Приклад. Знайти максимальне із двох дійсних чисел.

З використанням умовного оператора (?:)	**З використанням умовного оператора if**

`max = (x>y) ? x : y;`	`if (x>y)`
	`  max=x;`
	`else`
	`  max=y;`

6.4. Оператор вибору `switch`

Оператор `switch` дозволяє передавати керування одному з декількох операторів в залежності від значення виразу, який називають *селектором*

вибору. У якості селектора вибору може бути вираз цілого типу, типу `char`, переліковного типу або типу `string`.

Загальне правило запису	**Приклад** Вводиться оцінка – цифра, вивести оцінку прописом (селектор вибору цілого типу).

switch (<селектор вибору>)
{
case <константа1> : <оператор1>;
break;
case <константа2> : <оператор2>;
break;
. . . . . .
case <константаN> : <операторN>;
break;
default : <оператор N+1>;
break;
}
	using System;
	using System.Collections.Generic;
	using System.Linq;
	using System.Text;
	namespace ConsoleApplication1
	{
	class Program
	{
	static void Main(string[] args)
	{
	int mark;
	Console.Write("Mark = ");
	mark = int.Parse(Console.ReadLine());
	switch (mark)
	{
	case 2: Console.WriteLine("Незадовільно");
	break;
	case 3: Console.WriteLine("Задовільно.");
	break;
	case 4: Console.WriteLine("Добре");
	break;
	case 5: Console.WriteLine("Відмінно");
	break;
	default: Console.WriteLine("Неправильна оцінка.");
	break;
	}
	Console.ReadKey();
	}
	}
	}

Якщо для декількох констант вибору необхідно виконати одні і ті ж дії, тоді ці константи вибору записують послідовно одна за одною, після чого вказується загальна дія.

Приклад. З клавіатури вводиться оцінка у національній шкалі, необхідно вивести повідомлення про те, чи зараховано студенту залік.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int mark;
```

```
                Console.Write("Mark = ");
                mark = int.Parse(Console.ReadLine());
                switch (mark)
                {
                    case 1:
                    case 2:Console.WriteLine("Незараховано.");
                        break;
                    case 3:
                    case 4:
                    case 5:Console.WriteLine("Зараховано.");
                        break;
                    default: Console.WriteLine("Неправильна оцінка.");
                        break;
                }
                Console.ReadKey();
            }
        }
    }
```

Приклад. З клавіатури вводиться колір помідора, на екран необхідно вивести у якому він стані (росте, дозріває чи можна зірвати). При розв'язанні цього завдання використаємо оператор `switch`, у якому селектор і константи вибору є величинами типу `string`.

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    namespace ConsoleApplication1
    {
        class Program
        {
            static void Main(string[] args)
            {
                string color;
                Console.Write("Color = ");
                color = Console.ReadLine();
                switch (color)
                {
                    case "Green" : Console.WriteLine("Ще росте");
                            break;
                    case "Yellow": Console.WriteLine("Дозріває.");
                            break;
                    case "Red": Console.WriteLine("Можна зірвати");
                            break;
                    default: Console.WriteLine("Помідор-мутант.");
                              break;
                }
                Console.ReadKey();
            }
        }
    }
```

Приклад. З клавіатури вводиться буква у нижньому регістрі, з'ясувати, чи є буква голосною. При розв'язанні цього завдання використаємо оператор `switch`, у якому селектор вибору та константи вибору типу `char`.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            char c;
            Console.Write("Input letter= ");
            c = (char)Console.Read();
            switch (c)
            {
                case 'a' :
                case 'o' :
                case 'y' :
                case 'и' :
                case 'i' :
                case 'e' : Console.WriteLine("Голосна");
                           break;
                default: Console.WriteLine("Приголосна");
                    break;
            }
            Console.ReadKey();
        }
    }
}
```

Питання для самоконтролю

1. *Що називають розгалуженням?*
2. *Які форми умовного оператора розрізняють у мові C#?*
3. *Який загальний вигляд умовного оператора у повній формі?*
4. *Який загальний вигляд умовного оператора у скороченій формі?*
5. *У яких випадках використовують складений оператор?*
6. *Який загальний вигляд умовного оператора (?:)?*
7. *Який загальний вигляд оператора вибору* `switch`*?*
8. *У яких випадках використовують оператор вибору* `switch`*?*

Завдання для самостійної роботи

Завдання 1

№ варіанту	Умова
1	Дано два дійсних числа: a, b. З'ясувати, чи належать ці числа інтервалу $[1;2]\cup(3;7)$.
2	Дано дійсні числа: a, b, c, d. З'ясувати, чи належать ці числа інтервалу $[1;2]\cap(c;d)$.
3	Дано дійсне число a. З'ясувати, чи належать це число інтервалу $(3;7)\cup[8;9)\cup(11;22.4)$.
4	Дано цілі число $a,b,c\in Z$. Визначити, чи належить a множині $Z=\{3,4,5,...,9\}\cup\{b,b+1,b+2,...,c\}$.
5	Дано цілі число $a,b,c\in Z$. Визначити, чи належить a множині $Z=\{3,4,5,...,9\}\cap\{b,b+1,b+2,...,c\}$.
6	Дано три дійсних числа: a, b, c. Знайти $\max(a,b)+(\min(b,c))^2$.
7	Дано три дійсних числа: a, b, c. Знайти $\min(a,b)+(\min(b,c))^2$.
8	Дано три дійсних числа: a, b, c. Знайти $\min(a,b)+(\max(b,c))^2$.
9	Дано три дійсних числа: a, b, c. Знайти $\max(a,b)+(\max(a,b)+\min(b,c))^2$.
10	Дано три дійсних числа: a, b, c. Знайти $(\min(a,b,c))^2$.
11	Дано дійсні числа x, y, z. Обчислити $\max(x+y+z, xyz, xy-z)$.
12	Дано два дійсних числа: x, y $(x\neq y)$. Менше з них піднести до квадрату, а більше замінити пів-сумою цих чисел.
13	Дано $a,b,c\in R$. Замінити найбільше значення нулем.
14	Визначити координатну чверть, в яку потрапляє точка із координатами x і y $(xy\neq 0)$.
15	Дано дійсні числа a, b, c. Знайти суму тих з них, які належать інтервалу $[x, y]$.
16	Дано трицифрове натуральне число n. Визначити суму двох деяких цифр цього числа, яка є найбільшою.
17	Дано два двоцифрові натуральні числа n і m. Знайти суму найменшої цифри в числі n та найбільшої цифри в числі m.
18	Перевірити, чи лежить коло $(x-a)^2+(y-b)^2=r^2$ повністю в середині кола $(x-a_1)^2+(y-b_1)^2=r_1^2$.
19	Дано числа b,c та x_1, x_2. Визначити, чи є числа b,c коефіцієнтами квадратного рівняння $x^2+bx+c=0$, де x_1, x_2 – розв'язки цього квадратного рівняння.
20	Дано три дійсних числа: a, b, c. Визначити, чи є вони членами арифметичної прогресії.

Завдання 2

№ варіанту	Умова	№ варіанту	Умова
1	$y=\begin{cases}\cos(A+C+N), \text{ якщо } A=C=N,\\ \cos(A\cdot C\cdot N), \text{ якщо } A<C=N,\\ \cos((A+C)\cdot N), \text{ якщо } A<C<N,\\ 0, \quad \text{в інших випадках.}\end{cases}$	2	$y=\begin{cases}\ln\lvert x\rvert-n, \text{ якщо } x<n,\\ \ln\lvert x\rvert-n, \text{ якщо } x=n,\\ \cos(nx), \text{ якщо } x>n.\end{cases}$

№ варіанту	Умова	№ варіанту	Умова
3	$y=\begin{cases}\cos(x^2+\ln(x)), \text{якщо } x^2+\ln(x)>0\\ \dfrac{1}{(x^2+\ln(x))}, \text{якщо } x^2+\ln(x),\\ \cos(x), \text{якщо } x^2+\ln(x).\end{cases}$	4	$y=\begin{cases}1, \text{якщо } a<b<c,\\ 2, \text{якщо } a=b=c,\\ 3, \text{якщо } b<a<c,\\ 4, \text{якщо } b<a<c,\\ 0, \text{в інших випадках}.\end{cases}$
5	$z=\begin{cases}ye^x, \text{ якщо } y<x,\\ y\cdot x, \text{якщо } y=x,\\ x\cdot e^y, \text{якщо } y>x.\end{cases}$	6	$y=\begin{cases}1, \text{якщо } 0\le x<5,\\ 2, \text{якщо } 5\le x<8,\\ 3, \quad \text{якщо } x<0,\\ 4, \quad \text{якщо } x\ge 8.\end{cases}$
7	$y=\begin{cases}1, \text{ якщо } 2x^2-x-3=0,\\ 2, \text{ якщо } 2x^2-x-3>0,\\ 0, \text{якщо } 2x^2-x-3<0.\end{cases}$	8	$y=\begin{cases}0, \text{ якщо } 0\le n<5,\\ 1, \text{ якщо } 5\le n<10,\\ 2, \text{ якщо } 10\le n<15,\\ 3, \text{в інших випадках}.\end{cases}$
9	$y=\begin{cases}e^t\cdot\sin(t), \text{якщо } t<0\\ e^t\cdot\cos(t), \text{якщо } t\ge 0\end{cases}$ $t=\begin{cases}\cos(x), \text{якщо } x<2,\\ \ln(x), \text{якщо } x\ge 2.\end{cases}$	10	$z=\begin{cases}\ln(x)-tg(x), 1\le x\le 3,\\ tg(x), 3<x\le 4,\\ 0,\end{cases}$
11	$y=\begin{cases}(\cos(x))^n, \text{якщо } n=1,\\ x^n, \quad \text{якщо } n=2,\\ n^x, \quad \text{якщо } n=3,\\ 0, \text{в інших випадках}.\end{cases}$	12	$y=\begin{cases}0, \text{ якщо } n=1,2,3,4,\\ 1, \text{ якщо } n=10,11,12,\\ 2, \text{ якщо } n=15,19,\\ 3, \text{в інших випадках}.\end{cases}$
13	$y=\begin{cases}1, \text{ якщо } 2x^2-3=0,\\ 2x, \text{ якщо } 2x^2-3>0,\\ x, \text{якщо } 2x^2-3<0.\end{cases}$	14	$z=\begin{cases}(\sin(x))^n, \text{якщо } n=10,\\ x^n, \quad \text{якщо } n=22,\\ n^x, \quad \text{якщо } n=3,\\ 1, \text{в інших випадках}.\end{cases}$
15	$y=\begin{cases}x^2-3x+8, \text{ якщо } x^2-x\le 1,\\ \dfrac{1}{x^3-3x+8}, \text{якщо } x^2-x>1.\end{cases}$	16	$y=\begin{cases}-4, \text{ якщо } x<0,\\ x^2+3x+4, \text{якщо } 0\le x<1,\\ (x^2+3x+4)^2-1, \text{якщо } x\ge 1.\end{cases}$
17	$y=\begin{cases}0, \text{ якщо } x\le 0,\\ x^2-x, \text{ якщо } 0<x\le 1,\\ x^2-\sin\pi x^2, \text{ якщо } x>1.\end{cases}$	18	$y=\begin{cases}ax+by, \text{якщо } ax+by\in[c,d),\\ x+y, \text{ якщо } ax+by<c,\\ 1-x-y, \text{ якщо } ax+by\ge d.\end{cases}$ де a,b,c,d - відомі величини
19	$y=\begin{cases}e^{x^3+1}, \text{ якщо } x+y<0,\\ x^2y, \text{ якщо } x+y=0,\\ 0, \text{ якщо } x+y>0.\end{cases}$	20	$y=\begin{cases}arctg\dfrac{y}{x}, x>0, y\ge 0,\\ \dfrac{\pi}{2}, x=0, y>0,\\ \pi+arctg\dfrac{y}{x}, x<0,\\ \dfrac{3\pi}{2}, x=0, y<0,\\ 2\pi+arctg\dfrac{y}{x}, x>0, y<0\end{cases}$

Завдання 3

№ варіанту	Умова
1	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Визначити, чи є цей трикутник виродженим.
2	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Визначити, чи є цей трикутник прямокутним.
3	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Визначити, чи є цей трикутник гострокутним.
4	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Визначити, чи є цей трикутник тупокутним.
5	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Визначити, чи є цей трикутник рівнобедреним.
6	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Визначити, чи є цей трикутник рівностороннім.
7	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Знайти периметр трикутника.
8	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Знайти найбільшу сторону.
9	Дано чотири точки, що є вершинами чотирикутника $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ $D(x_4; y_4)$. З'ясувати, чи можуть вони бути вершинами паралелограма.
10	Дано чотири точки, що є вершинами чотирикутника $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ $D(x_4; y_4)$. З'ясувати, чи можуть вони бути вершинами ромба.
11	Дано чотири точки, що є вершинами чотирикутника $A(x_1; y_1; z_1)$, $B(x_2; y_2; z_2)$, $C(x_3; y_3; z_3)$ $D(x_4; y_4; z_4)$. З'ясувати, чи можуть вони бути вершинами паралелограма.
12	Дано чотири точки, що є вершинами чотирикутника $A(x_1; y_1; z_1)$, $B(x_2; y_2; z_2)$, $C(x_3; y_3; z_3)$ $D(x_4; y_4; z_4)$. З'ясувати, чи можуть вони бути вершинами ромба.
13	Дано чотири точки, що є вершинами чотирикутника $A(x_1; y_1; z_1)$, $B(x_2; y_2; z_2)$, $C(x_3; y_3; z_3)$ $D(x_4; y_4; z_4)$. З'ясувати, чи можуть вони бути вершинами прямокутника.
14	Дано чотири точки, що є вершинами чотирикутника $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ $D(x_4; y_4)$. З'ясувати, чи можуть вони бути вершинами трапеції
15	Дано три точки $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$. Визначити дві із трьох точок, які можуть утворити прямокутник найбільшої площі.
16	Дано три числа. З'ясувати, чи можуть вони бути вершинами різностороннього тупокутного трикутника.
17	Дано три числа. З'ясувати, чи можуть вони бути вершинами різностороннього гострокутного трикутника.
18	Дано три числа. З'ясувати, чи можуть вони бути вершинами рівнобедреного тупокутного трикутника.
19	Дано три числа. З'ясувати, чи можуть вони бути вершинами рівнобедреного гострокутного трикутника.
20	Дано три точки $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$. З'ясувати, чи можуть вони бути вершинами трикутника.

7. РЕАЛІЗАЦІЯ ЦИКЛІЧНИХ АЛГОРИТМІВ

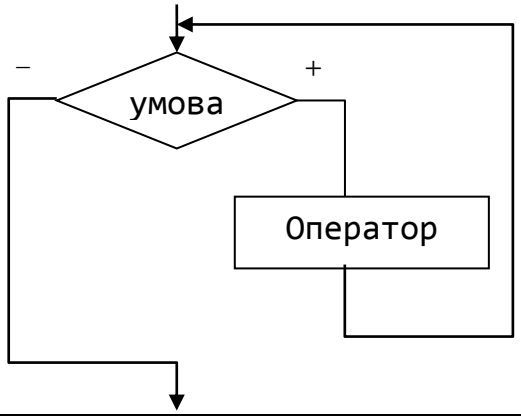
7.1. Оператори циклу

Часто постає потреба виконати один і той самий оператор декілька разів. для цього застосовують *оператори циклів*. Цикл складається із *заголовка* і *тіла*. У заголовку циклу зазначається умова завершення циклу, а тіло циклу являє собою оператор, який потрібно виконати декілька разів. Кожне виконання оператора тіла циклу називається його *ітерацією*.

7.1.1. Оператор циклу `while`

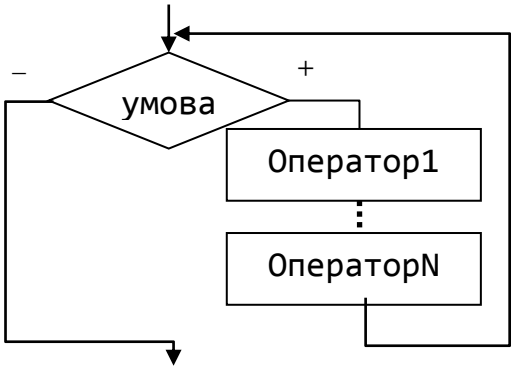
Оператор `while` циклічно виконує своє тіло до тих пір, поки умова виконується (логічний вираз приймає значення `true`).

Програмна структура	**Аналог на мові блок-схем**	**Приклад.** Знайти суму перших n натуральних чисел.

`while (<умова>)`
`  <оператор>;`
		`int sum=0;`
		`int i=1;`
		`while(i<=n)`
		`  sum=sum+i++;`

Якщо тіло циклу складається з більше ніж одного оператора, то необхідно використати складений оператор.

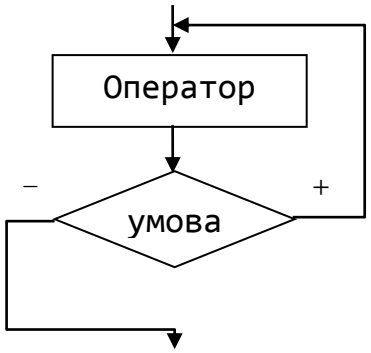
Програмна структура	**Аналог на мові блок-схем**	**Приклад.** Знайти суму і добуток перших n натуральних чисел.

`while (<умова>)`
`{`
`  <оператор1>;`
`  . . . . . .`
`  <операторN>;`
`}`
		`int sum=0;`
		`int mult=1;`
		`int i=1;`
		`while(i<=n)`
		`{`
		`  sum=sum+(i++);`
		`  mult=mult*(i++);`
		`}`

7.1.2. Оператор циклу `do-while`

Оператор циклу `do-while` відрізняється від оператора `while` тим, що перевірка умови виконується не до, а після виконання інструкції (оператора). У операторі циклу `do-while` тіло циклу виконається принаймні один раз.

Програмна структура	**Аналог на мові блок-схем**	**Приклад.** Знайти суму перших n натуральних чисел.

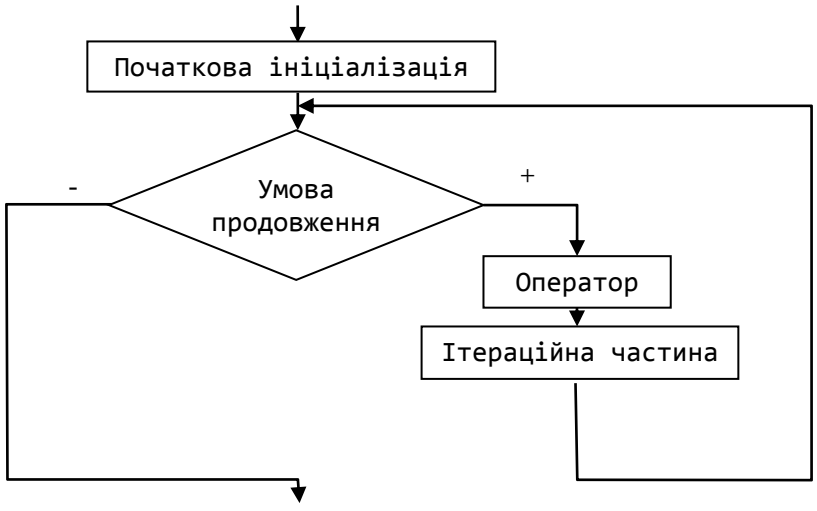
`do`
`{`
`  <оператор>;`
`}`
`while (<умова>)`
		`int sum=0;`
		`int i=1;`
		`do`
		`{`
		`  sum=sum+i++;`
		`}`
		`while(i<=n)`

7.1.3. Оператор циклу `for`

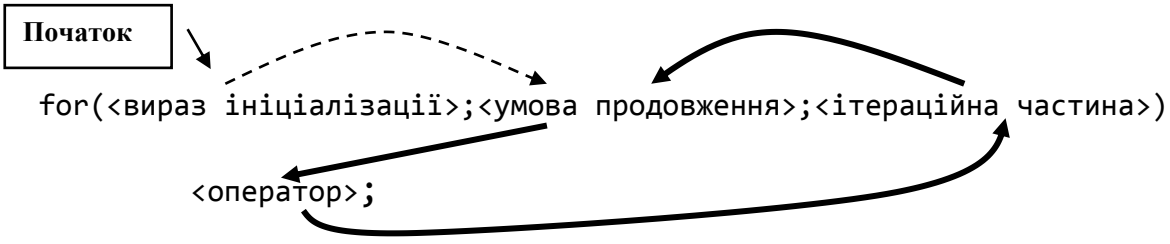
Загальний вигляд:

```
for (<вираз ініціалізації>;<умова продовження>;<ітераційна частина>)
      <оператор>;
```

Аналог на мові блок-схем:



Схематичне зображення виконання оператора:



Оператор `for` працює у відповідності до наступного алгоритму:

1. Обчислюється вираз ініціалізації. У цій частині допустима ініціалізація декількох лічильників циклу.
2. Перевіряється умова продовження. Якщо умова невірна то робота циклу завершується і передається управління наступному оператору.
3. Якщо умова істинна, виконується тіло даного оператора.
4. Виконується приріст одного або декількох лічильників циклу (або виконується довільна інша операція).
5. Здійснюється перехід до кроку 2.

Приклад. Знайти суму перших n натуральних чисел.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int n,sum=0;
            Console.WriteLine("n=");
            n=int.Parse(Console.ReadLine());
            for (int i = 1; i <= n; i++)
                sum = sum + i;
                Console.ReadKey();
        }
    }
}
```

Проілюструємо випадок, коли частина ініціалізації та ітераційна частина містять декілька виразів, розділених комою.

Приклад. Обчислити значення виразу

$$\frac{1}{0,3}+\frac{2}{0,4}+\ldots+\frac{n}{0,3+0.1*(n-1)}.$$

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
```

```
        class Program
        {
            static void Main(string[] args)
            {
                int n;
                Console.WriteLine("n=");
                n=int.Parse(Console.ReadLine());
                double d;
                int c;
                double sum=0;

                for (c = 1, d=0.3; c <= n; c++, d+=0.1)
                   sum +=c/d;

                Console.WriteLine("sum={0:f3}",sum);
                Console.ReadKey();
            }
        }
    }
```

Кожна з частин оператора циклу може бути відсутньою, але розділові знаки “;” є обов’язковими. Так нескінчений цикл може бути задано так:

```
for (  ;  ;  )
      <оператор>;
```

7.2. Оператори `break` та `continue`

Оператор `continue` може бути використаний у будь-якому із циклів у випадку, коли немає потреби виконувати до кінця усі оператори тіла циклу поточної ітерації, а необхідно одразу перейти до наступної ітерації.

Приклад. Знайти добуток непарних натуральних чисел, що менші за K.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int K;
            Console.Write("K=");
            K=int.Parse(Console.ReadLine());
            int mult=1;
```

```
                for (int i = 1;i<=K; i++)
                {
                    if ((i % 2) == 0) continue;
                    mult *= i;
                }

                Console.ReadKey();
            }
        }
    }
```

Оператор `break` у циклах використовують для негайного виходу з того циклу, в тілі якого він знаходиться.

Приклад. Знайти найменше значення факторіалу натурального числа, що перевищує число K.

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    namespace ConsoleApplication1
    {
        class Program
        {
            static void Main(string[] args)
            {
                int K;
                Console.Write("K=");
                K=int.Parse(Console.ReadLine());
                int fakt=1;
                for (int i = 1; ; i++)
                {
                    fakt *= i;
                    if (fakt > K) break;
                }
                Console.WriteLine("fakt={0}",fakt);
                Console.ReadKey();
            }
        }
    }
```

Приклад. Вивести на екран усі натуральні двоцифрові числа, у яких друга цифра не перевищує першу.

При розв’язанні цього завдання використаємо оператор `break` , який здійснює вихід тільки із вкладеного циклу (робота зовнішнього циклу продовжується).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 1; i <= 9; i++)
            {
                for (int j = 1; j <= 9; j++)
                {
                    if (j > i) break;
                    Console.WriteLine("{0}{1}",i,j);
                }
            }
                Console.ReadKey();

        }
    }
}
```

Питання для самоконтролю

1. *У яких випадках застосовують оператори циклу?*
2. *Що таке тіло циклу?*
3. *Що називають ітерацією циклу?*
4. *Який загальний вигляд оператора циклу* `while`*?*
5. *Який загальний вигляд оператора циклу* do-`while`*?*
6. *Які відмінності у роботі операторів циклу* `while` *ma* `do-while`*?*
7. *Який загальний вигляд оператора* `for`*?*
8. *За яким алгоритмом виконується оператор* `for`*?*
9. *Яке призначення оператора* `break`*?*
10. *Яке призначення оператора* `continue`*?*

Завдання для самостійної роботи

Завдання 1

№ варіанту	Умова
1	Дано $a \in R, n \in N$. Знайти $\frac{2}{a} + \frac{4}{a(a+1)} + ... + \frac{2n}{a(a+1)...(a+n)}$.
2	Дано $a \in R, n \in N$. Знайти $\frac{1}{a} + \frac{1}{a(a+1)} + ... + \frac{1}{a(a+1)...(a+n)}$.
3	Дано $a \in R$, $n \in N$. Знайти $\underbrace{\frac{1!}{a^2} + \frac{2!}{a^3(a+1)} + \frac{3!}{a^4(a+1)^2(a+2)} + \frac{4!}{a^5(a+1)^3(a+2)^2(a+3)} ...}_{n \text{ доданків}}$
4	Дано дійсне число a і натуральне число n. Обчислити $a(a+1)....(a+n-1)$.
5	Дано дійсне число a і натуральне число n. Обчислити $\ln \lvert a^n \rvert + \ln \lvert a^{n-1} \rvert + ... + \ln \lvert a \rvert$.
6	Дано дійсне число a і натуральне число n. Обчислити $a^2(a+1)^2....(a+n-1)^2$.
7	Дано $x, a \in R$, $n \in N$. Знайти $\underbrace{((...((}_{n} x+a)^2 + a)^2 + ... + a)^2 + a)^2 + a$.
8	Дано натуральне число n. Обчислити $s = \prod_{i=1}^{n} i^{i-1}$.
9	Дано $a_0 = 4$. Обчислити a_{12}, якщо $a_i = a_{i-1} - \cos(a^i / 12)$.
10	Дано натуральне число n. Обчислити $s = \left(2^2 + 4^2 + ... + (2n)^2\right) + \left(3^3 + 5^3 + ... + (2n+1)^3\right)$.
11	Дано натуральне число n і дійсне число x. Обчислити $\cos x + \cos^2 x + ... + \cos^n x$.
12	Дано $x \in R$, $n \in N$. Знайти $\sin x + \sin^2 x + ... + \sin^n x$.
13	Дано натуральне число n. Обчислити $\prod_{i=1}^{n} \frac{i-2}{i+1}$.
14	Дано натуральне число n. Обчислити $\sum_{i=1}^{n} \frac{1}{i!}$.
15	Задано натуральне число n. Чи утворюють цифри цього числа зростаючу послідовність.
16	Задано натуральне число n. Знайти кількість цифр числа n, які дорівнюють 1.
17	Задано натуральне число n. Знайти кількість цифр числа n.
18	Задано натуральне число n. Знайти суму цифр числа n.
19	Задано натуральне число n. Знайти кількість цифр числа n, які дорівнюють 1.
20	Задано натуральне число n. Знайти добуток парних цифр числа n.

Завдання 2

Обчислити з точністю $\varepsilon > 0$ значення функції, заданої за допомогою ряду Тейлора у деякій точці x.

№ варіанту	Функція
1	$\ln\frac{x+1}{x-1}=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+!}}=2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots\right) \quad \lvert x\rvert>1$
2	$e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\ldots, \ \lvert x\rvert<\infty$
3	$e^{-x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-\ldots \quad \lvert x\rvert<\infty$
4	$\ln(x+1)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1}=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\ldots \quad \lvert x\rvert<1$
5	$\ln\frac{1+x}{1-x}=2\sum_{n=0}^{\infty}\frac{x^{2n+!}}{2n+1}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots\right) \quad \lvert x\rvert<1$
6	$\ln(1-x)=-\sum_{n=1}^{\infty}\frac{x^n}{n}=-x-\frac{x^2}{2}-\frac{x^4}{4}-\ldots \quad \lvert x\rvert<1$
7	$arcctg\, x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1}=\frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}+\cdots \quad \lvert x\rvert\le 1$
8	$arctg\, x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\cdots \quad x>1$
9	$arctg\, x=\sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1}=x-\frac{x^3}{3}+\frac{x^5}{5}-\cdots \quad \lvert x\rvert\le 1$
10	$Arth\, x=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=x+\frac{x^3}{3}+\frac{x^5}{5}+\cdots \quad \lvert x\rvert<1$
11	$Arth\, x=\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\cdots \quad x>1$
12	$arctg\, x=-\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=-\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\cdots \quad x<-1$
13	$e^{-x^2}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{n!}=1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-\ldots \quad \lvert x\rvert<\infty$
14	$\cos(x)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-\ldots \quad \lvert x\rvert<\infty$
15	$\frac{\sin x}{x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)!}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+\ldots, \ \lvert x\rvert<\infty$
16	$\ln x=2\sum_{n=0}^{\infty}\frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}=2\left(\frac{x-1}{x+1}+\frac{(x-1)^3}{3(x+1)^3}+\frac{(x-1)^5}{5(x+1)^5}+\ldots\right), \ x>0$
17	$\ln x=\sum_{n=0}^{\infty}\frac{(x-1)^{n+1}}{(n+1)x^{n+1}}=\frac{x-1}{x}+\frac{(x-1)^2}{2x^2}+\frac{(x-1)^3}{3x^3}+\ldots, \ x>\frac{1}{2}$
18	$\ln x=\sum_{n=0}^{\infty}\frac{(-1)^n(x-1)^{n+1}}{(n+1)}=(x-1)-\frac{(x-1)^2}{2}+\frac{(x-1)^3}{3}-\ldots, \quad 0<x<2$
19	$\arcsin x=x+\sum_{n=1}^{\infty}\frac{1\cdot 3\cdots(2n-1)\cdot x^{2n+1}}{2\cdot 4\cdots 2n\cdot(2n+1)}=x+\frac{x^3}{2\cdot 3}+\frac{1\cdot 3\cdot x^5}{2\cdot 4\cdot 5}+\frac{1\cdot 3\cdot 5\cdot x^7}{2\cdot 4\cdot 6\cdot 7}+\frac{1\cdot 3\cdot 5\cdot 7\cdot x^9}{2\cdot 4\cdot 6\cdot 8\cdot 9}\ldots, \ \lvert x\rvert<1$
20	$\arccos x=\frac{\pi}{2}-\left(x+\sum_{n=1}^{\infty}\frac{1\cdot 3\cdots(2n-1)\cdot x^{2n+1}}{2\cdot 4\cdots 2n\cdot(2n+1)}\right)==\frac{\pi}{2}-\left(x+\frac{x^3}{2\cdot 3}+\frac{1\cdot 3\cdot x^5}{2\cdot 4\cdot 5}+\frac{1\cdot 3\cdot 5\cdot x^7}{2\cdot 4\cdot 6\cdot 7}+\ldots\right), \ \lvert x\rvert<1$

8. СТРУКТУРНИЙ ПІДХІД ДО РОЗРОБКИ АЛГОРИТМІВ. ФУНКЦІЇ

При розв'язанні складних задач застосовується структурний підхід до розробки алгоритмів за методикою «зверху-вниз». Цей підхід полягає у розбитті складної задачі на більш прості підзадачі.

8.1. Означення функції

У мові C# у рамках структурного підходу до розробки алгоритмів підзадачі реалізовують у вигляді окремих функцій, які потім можуть бути використані в основному методі програми `Main`. Під функцією будемо розуміти окремо виділений програмний блок, що має власне ім'я та може повертати результат у викликаючу програму.

Загальна форма	**Приклад.** Знайти максимальне з трьох цілих чисел

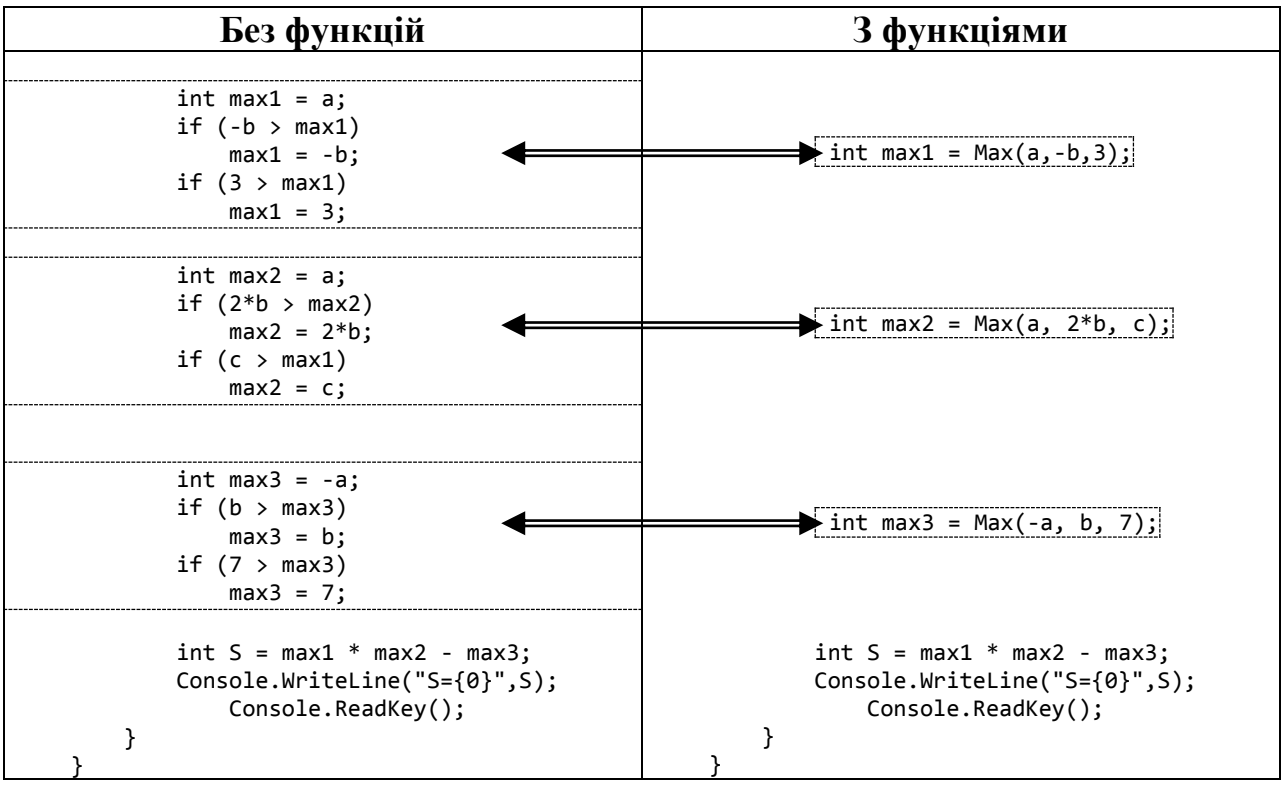
заголовок функції
`static <тип результату>`
`<ім'я функції>(<список форм. парам.>)`
`{`
`  //тіло функції`
`}`
	`static int Max(int c1,int c2,int c3)`
	`  {`
	`  int m = c1;`
	`  if (c2 > m)`
	`  m = c2;`
	`  if (c3 > m)`
	`  m = c3;`
	`  return m;`
	`  }`

Проілюструємо застосування структурного підходу на прикладі. Обчислити значення виразу

$$S = \underbrace{\max\{a,-b,3\}}_{\max 1} * \underbrace{\max\{a,2b,c\}}_{\max 2} - \underbrace{\max\{-a,b,7\}}_{\max 3} = \max 1 * \max 2 - \max 3$$

Без функцій	**З функціями**

`class Program`
`{`
 	
`  static void Main(string[] args)`
`  {`
`  int a,b,c;`
`  Console.Write("a=");`
`  a = int.Parse(Console.ReadLine());`
`  Console.Write("b=");`
`  b = int.Parse(Console.ReadLine());`
`  Console.Write("c=");`
`  c = int.Parse(Console.ReadLine());`
	`class Program`
	`{`
	`  static int Max(int c1, int c2, int c3)`
	`  {  int m = c1;`
	`  if (c2 > m)`
	`  m = c2;`
	`  if (c3 > m)`
	`  m = c3;`
	`  return m;`
	`  }`
	
	`  static void Main(string[] args)`
	`  {`
	`  int a,b,c;`
	`  Console.Write("a=");`
	`  a = int.Parse(Console.ReadLine());`
	`  Console.Write("b=");`
	`  b = int.Parse(Console.ReadLine());`
	`  Console.Write("c=");`
	`  c = int.Parse(Console.ReadLine());`



Якщо функція не повертає якого-небудь результату, тоді в якості типу результату вказується тип `void`. Для повернення результату роботи функції використовується інструкція `return`.

Загальне правило:

`return [<вираз того ж типу що й результат функції>];`

Крім того, ключове слово `return` завершує виконання функції. Тобто, якщо функція не повертає якого-небудь результату, то інструкцію `return` без виразу також можна використовувати для завершення виконання функції. Якщо ключове слово `return` відсутнє, то виконання функції завершиться, коли буде досягнуто кінець блоку функції. Якщо тип результату функції відмінний від `void`, то використання ключового слова `return` є обов'язковим.

8.2. Формальні та фактичні параметри

Параметри, що їх імена вказані у `<список форм. парам.>` у заголовку функції, називаються *формальними*. Формальні параметри в списку формальних параметрів розділяються комою. В якості специфікаторів формальних параметрів можуть використовуватися ключові слова **`ref`** та **`out`**.

Оскільки специфікатор формального параметра є частиною списку формальних параметрів, то можна описувати функції, які відрізняються специфікаторами формальних параметрів.

Значення, що замінюють формальні параметри під час виклику функції, називаються *фактичними параметрами* або *аргументами*.

Описання функції створює окрему область дії імен для її формальних параметрів та локальних (по відношенню до функції) змінних, які визначені в блоці, який утворює тіло методу. В межах блоку звернення до формальних параметрів здійснюється за іменами, визначеними в списку формальних параметрів.

При виклику функції створюється окрема копія формальних параметрів та локальних змінних даної функції, після чого відбувається передача аргументів у копії. В залежності від специфікатора, який був використаний при описі формального параметра, аргументи можуть передаватися:

- за значенням (немає ніякого специфікатора);
- за покажчиком (вказано специфікатор `ref` або `out`).

При передачі аргументу *за значенням* в копію відповідного формального параметру просто записується значення, яке передається в даному аргументі. Копія розглядається як локальна змінна, яка має тип відповідного формального параметра. Зміна значення цієї змінної ніяк не вплине на значення фактичного параметра.

Приклад. Створити функцію, що повертає середнє арифметичне значення трьох дійсних чисел.

```
class Program
{
    static double Average(double c1, double c2, double c3)
    {
        return (c1 + c2 + c3) / 3;
    }
    static void Main(string[] args)
    {
        double n, m, k;
        Console.WriteLine("n=");
        n = double.Parse(Console.ReadLine());
        Console.WriteLine("m=");
```

```
            m = double.Parse(Console.ReadLine());
            Console.WriteLine("k=");
            k = double.Parse(Console.ReadLine());
            double ser= Average(n, m, k);
            Console.WriteLine("Average = {0} ",ser);
            Console.ReadKey();
        }
    }
```

При передачі аргументу за покажчиком у копію формального параметру записується адреса на даний аргумент. Тому, всі зміни стосовно даного формального параметра стосуються також і відповідного фактичного параметра. При описанні відповідного формального параметра та при передачі відповідного аргументу використовується модифікатор `ref`.

Такий спосіб передачі параметрів використовується, напиклад, тоді, коли в функції необхідно змінити фактичні параметри.

Приклад. Створити функцію, яка більше значення зменшує на 2, а менше збільшує на 3.

```
class Program
{
    static void Fun1(ref int a, ref int b)
    {
        if (a > b)
        {
            a -= 2;
            b += 3;
        }
        else
        {
            b -= 2;
            a += 3;
        }
    }
    static void Main(string[] args)
    {
        int n,m;
        Console.WriteLine("n=");
        n=int.Parse(Console.ReadLine());
        Console.WriteLine("m=");
        m = int.Parse(Console.ReadLine());
        Fun1(ref n,ref m);
        Console.WriteLine("n= {0} m={1}",n,m);
        Console.ReadKey();
    }
}
```

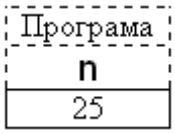
Для передачі аргументу за покажчиком можна використовувати специфікатор `out.` У цьому випадку, на відміну від попереднього, вже не є обов'язковою ініціалізація аргументу до його передачі у функцію. Але в самому методі фактичний параметр вважається неініціалізованим і має обов'язково бути проініціалізованим до завершення методу. При передачі аргументу у функцію також використовується специфікатор `out`.

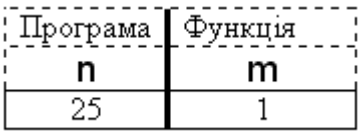
Такий метод передачі параметрів використовується, наприклад, тоді, коли аргумент є результатом роботи функції.

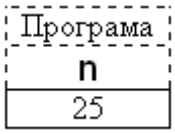
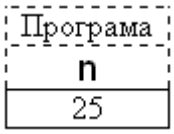
Приклад. Створити функцію, яка повертає максимальне та мінімальне значення із двох дійсних чисел.

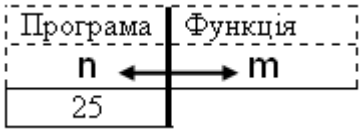
```
class Program
{
 static void MinMax(int c1, int c2, out int max, out int min)
    {
        if (c1 > c2)
        {
            max = c1;
            min = c2;
        }
        else
        {
            max = c2;
            min = c1;
        }
    }
    static void Main(string[] args)
    {
        int n,m;
        Console.WriteLine("n=");
        n=int.Parse(Console.ReadLine());
        Console.WriteLine("m=");
        m = int.Parse(Console.ReadLine());
        int max, min;
        MinMax(n, m, out max, out min);
        Console.WriteLine("max= {0} min={1}",max,min);
        Console.ReadKey();
    }
}
```

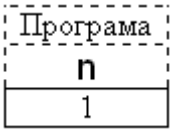
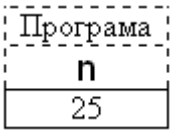
Отже, зміна значення фактичного параметра в результаті зміни значення формального параметра відбувається тільки у випадку коли передача здійснюється за покажчиком.

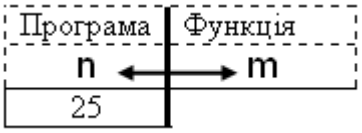
<table>
<tr><th>Передача за значенням
(немхє ніяких специфікаторів)
(інціалізація змінної n
обов’язкова)</th><th>Передача за покажчиком
(специфікатор ref)
(інціалізація змінної n
обов’язкова)</th><th>Аргумент як вихідне значення
(специфікатор out)
(інціалізація змінної n не
обов’язкова)</th></tr>
<tr><td><pre>class Program
{
 static void Fun1(int m)
 {
   m = 1;
 }
static void Main(string[] args)
{
    int n = 25;
    Fun1(n);
  Console.WriteLine("n= {0}",n);
  Console.ReadKey();
 }
}</pre></td><td><pre>class Program
{
 static void Fun1(ref int m)
 {
   m = 1;
 }
static void Main(string[] args)
{
    int n = 25;
    Fun1(ref n);
  Console.WriteLine("n= {0}",n);
  Console.ReadKey();
 }
}</pre></td><td><pre>class Program
{
 static void Fun1(out int m)
 {
   m = 1;
 }
static void Main(string[] args)
{
    int n;
    Fun1(out n);
  Console.WriteLine("n= {0}",n);
  Console.ReadKey();
 }
}</pre></td></tr>
<tr><td>Вивід програми:
n=25</td><td>Вивід програми:
n=1</td><td>Вивід програми:
n=1</td></tr>
<tr><td>Змінні в пам’яті ЕОМ:
До виклику:
 

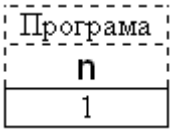
Під час роботи функції:
(для змінної m виділяється нова пам’ять)
 

Після роботи функції:
 
</td><td>Змінні в пам’яті ЕОМ:
До виклику:
 

Під час роботи функції:
(для змінної m нова пам’ять не виділяється а використовується пам’ять змінної n)
 

Після роботи функції:
 
</td><td>Змінні в пам’яті ЕОМ:
До виклику:
 

Під час роботи функції:
(для змінної m нова пам’ять не виділяється а використовується пам’ять змінної n)
 

Після роботи функції:
 
</td></tr>
</table>

8.3. Рекурсія

Рекурсія − це такий спосіб організації обчислювального процесу, за якого функція звертається сама до себе. Такі звернення називаються *рекурсивними викликами*, а функція, що містить рекурсивні виклики, − *рекурсивною*.

Рекурсію використовують у ситуаціях, коли легко звести вихідну задачу до задачі того ж виду, але з іншими вихідними даними. Найпростішим прикладом такої задачі може стати обчислення факторіала. Так обчислення факторіала може бути здійснене у відповідності до наступного рекурентного правила:

$$n!=\begin{cases}(n-1)!\cdot n, \text{ якщо } n>0;\\ 1, \text{ якщо } n=0.\end{cases}$$

Реалізуємо це рекурентне правило у вигляді рекурсивної функції:

```
static int factorial(int n)
{
    if (n == 0)
        return 1;
    else
        return factorial(n - 1) * n;
}
```

Будь-яке рекурентне правило, яке визначає деяку рекурсію повинно містити умову зупинення, що називається *умовою завершення рекурсії*. Для випадку задачі обчислення факторіала *n*! умовою завершення рекурсії є умова *n*=0.

Роботу рекурсивної функції розглянемо на прикладі обчислення 4!. Основна програма звертається до функції `factorial(4)`. Виклик функції `factorial(3)` відбудеться ще до першого виконання операції множення, після цього, теж ще до операції множення, з функції `factorial(3)` буде викликана функція `factorial(2)`. Процес рекурсивних викликів припиниться у функції `factorial(0)`, що поверне 1. Ця одиниця стане значенням лівого операнда операції `factorial(0)*1` у функції `factorial(1)`. Результат операції буде повернутий функцією `factorial(1)` у функцію `factorial(2)` і т.д. Отже, під час рекурсивних викликів функції `factorial` виконання операцій множення відкладатиметься, оскільки один із множників ще не відомий. Добутки будуть послідовно обчислені при поверненні із функцій.

Приклад. Описати рекурсивну функцію, яка за заданим дійсним значенням *x* та цілим значенням *n* обчислює значення величини x^n у відповідності до формули:

$$x^n=\begin{cases}1, \text{ якщо } n=0,\\ \dfrac{1}{x^{|n|}}, \text{ якщо } n<0,\\ x\cdot x^{n-1}, \text{ якщо } n>0.\end{cases}$$

```
        static double f(double x, int n)
        {
            if (n == 0)
                return 1;
            else
                if (n < 0)
                    return 1 / f(x, Math.Abs(n));
                else
                    return x * f(x, n - 1);
        }
```

Питання для самоконтролю

1. *У чому полягає суть структурного підходу до розробки алгоритмів?*
2. *Який загальний вигляд статичної функції?*
3. *Що таке формальні параметри?*
4. *Що таке фактичні параметри?*
5. *Які типи формальних параметрів існують у мові C#?*
6. *Що таке рекурсія?*

Завдання для самостійної роботи

Завдання 1

№ варіанту	Умова
1	За даними дійсними числами a і b обчислити $u = f(a,b) + f(2,a) + 2$, де $f(x,y) = \begin{cases} x^3 + \sqrt{x^2 + y^4}, x > y, \\ \dfrac{x^2 - 2x + \sqrt{x}}{\sqrt[5]{x^3}}, \text{інакше}. \end{cases}$
2	За даними дійсними числами a і b обчислити $u = f(a,b) + f(2,a) + 2$, де $f(x,y) = \begin{cases} x^3 + \sqrt{x^2 + y^4}, x > 0 \, і \, y > 0, \\ \dfrac{x^2 - 2x + \sqrt{x}}{\sqrt[5]{x^3}}, x > 0 \, і \, y < 0, \\ \sin(x * y), \text{інакше}. \end{cases}$
3	За даними дійсними числами a і b обчислити $u = f(a) + f(2) + f(b)$, де $f(x) = \begin{cases} \sum_{i=1}^{5} x^i, x > 0, \\ \prod_{i=1}^{5} x^i, \text{інакше}. \end{cases}$

№ варіанту	Умова
4	За даними дійсними числами a і b обчислити $u=\min\{f(a),f(b)\}$, де $f(x)=\begin{cases}\sin x+\sin x^2+...+\sin x^6, x>3,\\ -\cos x+\cos\cos x+...+(-1)^5\underbrace{\cos...\cos}_{5}x, \textit{інакше}.\end{cases}$
5	За даними дійсними числами a і b обчислити $u=\max\{f(a),f(b)\}$, де $f(x)=\begin{cases}\sin x+\sin^3 x+\sin^5 x+...+\sin^{17} x, x>3,\\ 15+tgx+tg\,tgx+...+\underbrace{tg...tg}_{5}x, \textit{інакше}.\end{cases}$
6	За даними дійсними числами a і b обчислити $u=\max\{f(a),f(b)\}-f(3)$, де $f(x)=\begin{cases}1+\frac{x}{2!}+\frac{x^2}{3!}+..+\frac{x^8}{8!}, x<0,\\ 15+\sqrt{\cos^6 x}, 0\le x\le 5,\\ \min\{1,2*\sin(x)\}, x>5.\end{cases}$
7	Дано дійсні числа x,y,z . Обчислити $u=\frac{\max\{x,z\}+\max\{x+y,xy\}}{\max^2\{x+y,xy\}}$.
8	Дано дійсні числа x,y,z . Обчислити $u=\frac{\max\{x,y,z\}+\max\{x+y,xy,4z\}}{\max\{\max^2\{x+y,xy,x^2\},7,z^2\}}$.
9	Використовуючи підпрограму для знаходження коренів квадратного рівняння, знайти розв'язок наступної системи рівнянь $\begin{cases}ax^2+2x+7=0,\\ bx^2-5x+2=0.\end{cases}$ Числа $a,b\in R$.
10	Використовуючи підпрограму для знаходження найбільшого спільного дільника (НСД), знайти значення виразу S=(НСД(a,b)+ НСД(a,4))+ НСД(24,b)
11	Використовуючи підпрограму для знаходження коренів квадратного рівняння, знайти розв'язок наступної системи рівнянь $\begin{cases}ax^2+2x+7=0,\\ bx^2-5x+2=0.\end{cases}$ Числа $a,b\in R$.
12	Дано три дійсних числа: a, b, c . Використовуючи підпрограми для знаходження максимальнго та мінімального серед двох дійсних чисел знайти $\max(a,b)+(\max(a,b)+\min(b,c))^2$.

№ варіанту	Умова
13	Обчислити значення виразу $f(2,a,4)-5*f(a,b,-c)$, де $f(x,y,z)=\begin{cases}\lg(x+y-z), & \text{якщо } x+y-z>10,\\ (\lvert x+y\rvert+1)^z, & \text{якщо } z-10<x+y<z+1,\\ x^2+y^2-z^3, & \text{якщо } x+y-z=1,\\ \cos^2 x+\sin y-e^{2z+1}, & \text{в інших випадках.}\end{cases}$
14	Обчислити значення виразу $f(4)+2f(a)-f(b)$, де $f(x)=\begin{cases}0, & x<23;\\ 1, & x=23;\\ -1, & x>23.\end{cases}$
15	Дано дійсні числа a,b,c,d . Обчислити $y=(p(a)+p(b)+p(c+2)+p(2d))/4$, де $p(x)=2x^2+\frac{3}{x}$.
16	Дано ціле число a. Визначити $3f(a/2)+5f(a^2-2)$, де $f(x)=\sum_{i=1}^{10}\frac{x^i}{i}$.
17	Дано дійсні числа x, y. Обчислити: $f(y,-2x,x+y-1.17)+f(2.2,y,x^2-y^2)$, де $f(a,b,c)=\frac{2a-b-\sin c}{5+\lvert c\rvert}$.
18	Дано дійсні числа x, y. Обчислити: $g(1.2,x)+g(y,x)-g(2x-1,xy)$, де $g(a,b)=a+2ab+3b+4$.
19	Дано ціле число a. Визначити: $12.5f(a/2)+3.1f(a^2+a+1)$, де $f(x)=\sum_{i=1}^{5}a^i$.
20	Дано дійсні числа x, y. Обчислити: $f(f(1,2),x)+f(y,f(x+y,f(2x,3y)))$ де $f(n,m)=n^2+2nm+3m^2+n+m$.

Завдання 2

№ варіанту	Умова
1	Використовуючи підпрограму для знаходження скалярного добутку, обчислити значення виразу s=2<a,b>-3<a,c>, де a,b,c$\in R^n$, <x,y> – скалярний добуток векторів.
2	Використовуючи підпрограму наближеного знаходження визначеного інтегралу за формулою лівих прямокутників, обчислити значення виразу $s=\int_a^3\sqrt{4x+\sin\sqrt{x^3}}\,dx+\int_a^b\sqrt{4x+\sin\sqrt{x^3}}\,dx$
3	Використовуючи підпрограму наближеного знаходження визначеного інтегралу за формулою правих прямокутників, обчислити значення виразу $s=\int_a^3\sqrt{4x+\sin\sqrt{x^3}}\,dx+\int_a^b\sqrt{4x+\sin\sqrt{x^3}}\,dx$

№ варіанту	Умова
4	Використовуючи підпрограми для додавання векторів та множення вектора на число, знайти вектор c=a-3*b+2*c, де a,b,c$\in R^n$.
5	Використовуючи підпрограму визначення паралельності двох прямих на площині, визначити, скільки взаємно паралельних пар прямих є серед вказаних n прямих: $A_i x + B_i y + C_i = 0,\ i = 1,2,...,n$.
6	Використовуючи підпрограму визначення перпендикулярності двох прямих на площині, визначити, скільки взаємно перпендикулярних пар прямих є серед вказаних n прямих: $A_i x + B_i y + C_i = 0,\ i = 1,2,...,n$.
7	Два трикутники задано координатами вершин. Використовуючи підпрограму визначення належності точки внутрішності трикутника, з'ясувати, чи лежить один з трикутників у середині іншого.
8	Трикутник задано координатами своїх вершин на площині. Використовуючи підпрограму для знаходження кута між векторами на площині, встановити тип трикутника (гострокутний, прямокутний, тупокутний).
9	Дано послідовність натуральних чисел $a_1, a_2, ..., a_n$. Використовуючи підпрограму, яка дозволяє встановити, чи є послідовність із чотирьох чисел арифметичною прогресією, знайти кількість послідовно розміщених четвірок чисел, які утворюють арифметичну прогресію.
10	Дано послідовність натуральних чисел $a_1, a_2, ..., a_n$. Використовуючи підпрограму, яка дозволяє встановити, чи є послідовність із чотирьох чисел геометричною прогресією, знайти кількість послідовно розміщених четвірок чисел, які утворюють геометричну прогресію.
11	Використовуючи відповідні підпрограми, з'ясувати, що є більшим, середнє арифметичне чи середнє геометричне чисел $a_1, a_2, ..., a_n$.
12	Дано послідовність натуральних чисел $a_1, a_2, ..., a_n$. Використовуючи відповідні підпрограми знаходження суми та добутку цифр, знайти натуральне число, у якого найменший добуток цифр.
13	Дано послідовність натуральних чисел $a_1, a_2, ..., a_n$. Використовуючи підпрограму знаходження найбільшої та найменшої цифри, знайти число, яке містить найбільшу цифру та число, яке містить найменшу цифру.
14	Дано послідовність з m натуральних чисел $n_1, n_2, ..., n_m$. Обчислити суму цифр всіх чисел.
15	Дано дійсні координати чотирьох точок на площині. Визначити, які трійки з них утворюють трикутники, і обчислити площу більшого трикутника, створивши для цього функції обчислення довжини відрізка та площі трикутника.
16	Дано послідовність з m натуральних чисел $n_1, n_2, ..., n_m$. Обчислити суму цифр всіх чисел, які більші за 5.
17	Дано послідовність з m натуральних чисел $n_1, n_2, ..., n_m$. Обчислити суму цифр, які кратні 3.
18	Дано послідовність з m натуральних чисел $n_1, n_2, ..., n_m$. Обчислити добуток цифр всіх чисел.
19	Дано послідовність з m натуральних чисел $n_1, n_2, ..., n_m$. Визначити число, добуток цифр якого є найменшим.

№ варіанту	Умова
20	Дано послідовність з m натуральних чисел $n_1, n_2, ..., n_m$. Визначити число, сума цифр якого є найбільшою.

Завдання 3

№ варіанту	Умова
1	
2	Використовуючи відповідну підпрограму знаходження g_i, обчислити значення виразу $s = g_7 + g_9$, де $g_i = \sin(g_{i-1} + \cos(g_{i-2}))$, $g_0 = 9, g_1 = 35$.
3	Нехай $x_0 = 1$, $x_i = x_{i-1} + 2i$, де $i = 1,2,...$. Визначити x_{10}.
4	Нехай $x_0 = x_1 = 1$, $x_i = x_{i-1} + 2x_{i-2}$, де $i = 2,3,...$. Визначити x_n.
5	Нехай $x_0 = x_1 = 1$, $x_i = x_{i-1} + x_{i-2}$. Визначити x_n.
6	Нехай $x_0 = 0, x_1 = 9$, $x_i = 2x_{i-1} + 3x_{i-2}$. Визначити x_n.
7	Нехай $x_0 = 0, x_1 = x_2 = 9$, $x_i = x_{i-1} + x_{i-2} + x_{i-3}$. Визначити x_n.
8	Нехай $x_0 = 0, x_1 = x_2 = 9$, $x_i = x_{i-1} + 4x_{i-3}$. Визначити x_n.
9	Нехай $x_0 = 0, x_1 = 7$, $x_i = x_{i-1}(1 + x_{i-2})$. Визначити x_n.
10	Нехай $x_0 = 0, x_1 = x_2 = x_3 = 7$, $x_i = \frac{x_{i-1}(1 + x_{i-2}) + x_{i-3}}{x_{i-4}}$. Визначити x_n.
11	Нехай $x_0 = 0, x_1 = x_2 = x_3 = 7$, $x_i = \frac{x_{i-1} + 4x_{i-2}(1 + x_{i-2}) + x_{i-3}}{\sqrt{x_{i-4}}} + x_{i-4}$. Визначити x_n.
12	Нехай $x_0 = x_1 = 1$, $x_i = \sin x_{i-1} + \frac{x_{i-2}}{\cos(x_{i-1})}$. Визначити x_n.
13	Нехай $x_0 = 2, x_1 = 1$, $x_i = \cos x_{i-1}^{x_{i-2}}$. Визначити x_n.
14	Нехай $x_0 = 2, x_1 = x_2 = 3$, $x_i = 7x_{i-1} - x_{i-2} * x_{i-3}$. Визначити x_n.
15	Дано цілі *m*, *n*. Обчислити: $A(n,m) = \begin{cases} m+1, n = 0, \\ A(n-1,1), n > 0, m = 0, \\ A(n-1, A(n, m-1)), \textit{інакше}. \end{cases}$
16	Дано цілі *m*, *n*. Обчислити: $\textit{нсд}(n,m) = \begin{cases} m, n = m, \\ \textit{нсд}(n-m, m), n > m, \\ \textit{нсд}(n, m-n), \textit{інакше}. \end{cases}$
17	Дано дійсні *m*, *n*. Обчислити: $S(n,m) = \begin{cases} 1, n = m = 1, \\ S(n-1, m) + 1, n > 1, \\ S(n, m-1), \textit{інакше}. \end{cases}$
18	Дано натуральне *n*. Обчислити: $f(n) = \begin{cases} 1, n = 1, \\ f(n-1) + n + n - 1, n > 1. \end{cases}$

№ варіанту	Умова
19	Дано натуральні *m*, *n*. Обчислити: $C(n,m)=\begin{cases}1, m=0, n>0 \text{ або } m=n\geq 0,\\ 0, m>n\geq 0,\\ C(n,-1m-1)+C(n-1,m), \text{інакше}.\end{cases}$
20	Напишіть рекурсивну функцію яка перевіряє чи однаково читається натуральне число зліва-направо.

ЛІТЕРАТУРА

1. *Павловская Т.А.* C#. Программирование на языке высокого уровня. Учебник для вузов / Т.А. Павловская – СПб.: Питер, 2007. – 432 с.
2. *Голуб Б.М.* C#. Концепція та синтаксис. Навч. посібник / Б.М. Голуб, – Львів: Видавничий центр ЛНУ імені Івана Франка, 2006. – 136 с.
3. *Троелсен Э.* Язык программирования C# 2010 и платформа .NET 4.0, 5-е изд. / Э Троелсен. – М.: ООО "И.Д. Вильямс", 2011. – 1392 с.
4. *Шилдт Г.* C# 4.0: полное руководство / Г. Шилдт. – М.: ООО "И.Д. Вильямс", 2011. – 1056 с.
5. *Караванова Т.П.* Інформатика: основи алгоритмізації та програмув.:777 задач з рек. та прикл.: Навч. посіб для 8-9 кл. із поглибл. вивч. інф-ки / Т.П. Караванова / За заг. ред.. М.З.Згуровського – К.:Ґенеза, 2006. – 286 с.

ЗМІСТ

Відповідальний за випуск: завідувач кафедрою системного аналізу і теорії оптимізації к.ф.-м.н., доц. Кузка О.І.

Автори: к. ф.-м. н., доц. Брила А.Ю.,
к. ф.-м. н., Антосяк П.П.,
к. ф.-м. н., Глебена М.І.,
к. ф.-м. н., доц. Чупов С.В.,
к. т. н., доц. Семйон І.В.

Рецензенти: к.ф.-м.н., доц. Мич І.А.,
к. ф.-м.н., доц. Погоріляк О.О.

ОСНОВИ ПРОГРАМУВАННЯ У C#

Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного факультету спеціальності "Прикладна математика"