Вапнічний С.Д., Зубик В.В., Ребрина В.А.

Факультативний курс з інформатики «Основи програмування. С++. Перший рік навчання»

для учнів 7-9 класів

загальноосвітніх навчальних закладів

Схвалено для використання в загальноосвітніх навчальних закладах області
Науково-методичною Радою Хмельницького ОІППО
(протокол № 2 від 22 червня 2010 року)

Хмельницький, 2010

Програма факультативного курсу з інформатики «Основи програмування С++» для учнів 7-9 класів

Пояснювальна записка

Факультатив «Основи програмування С++» розрахований на учнів 7-9 класів шкіл Хмельницької області.

Метою даного факультативу з інформатики **«Основи програмування С++»** є:

формування теоретичної бази знань учнів з основ програмування та практичних навичок розв'язування задач, які вже протягом 25 років пропонуються на олімпіадах з інформатики різного рівня починаючи від шкільних олімпіад закінчуючи міжнародними.

Активізувати роботу в області з метою підготовки учнів до участі у шкільних, районних та міських олімпіадах з інформатики.

Надати рівні можливості учням у підготовці до надзвичайно захоплюючої діяльності - програмування.

Особливістю даного факультативу є модульний принцип організації роботи факультативу. Він полягає у тому, що всі заняття будуть проводитися частинами (модулями) по 4 уроки. Перші два (одна субота) вчитель з учнями вивчають новий матеріал, закріплюють його, розв'язуючи задачі. Третій урок (наступна субота) - проводитиметься контроль вивченого через мережу Інтернет у реальному часі через контестер. Четвертий урок для узагальнення результатів проведеного контролю і проведення корекції знань учнів.

Розрахований на використання мови програмування С++ та середовища програмування **GCC**

Курс розрахований на **68** год.

Розподіл годин між темами факультативного курсу "Основи програмування":

№	Тема	Теоретичні	Практичні	Разом
1	Основні поняття мови програмування.	2	6	8
2	Логіка мови програмування.	2	6	8
3	Організація циклів.	2	6	8
4	Процедури. Формальні та фактичні параметри.	2	2	4
5	Функції.	2	2	4
6	Масиви.	2	10	12
7	Масиви символів, рядкові величини.	2	6	8
8	Рекурсивні функції та процедури.	2	2	4
9	Використання множин.	2	2	4
10	Робота з файлами даних.	2	2	4
11	Структури даних. Записи.	2	2	4
Всього				**68**

Тематичне планування

Тема 1: Основні поняття мови програмування. (8 годин)

Приклад простої програми. Константи та змінні. Оператор присвоювання. Арифметика Паскаля. Тип змінних. Арифметичні вирази. Операції. Функції. Ввід та вивід на консоль.

Учні повинні знати:

- принципи побудови, опису програми мовою програмування;
- основні типи даних у мові програмування;
- набір базових функцій та операцій у мові програмування;
- сутність операції присвоювання;
- сутність вказівок введення та виведення;

Учні повинні вміти:

- користуватися інтегрованим середовищем програмування;
- застосовувати вказівки введення та виведення, присвоювання при складанні лінійних програм;
- налагоджувати лінійні програми;
- відправляти розв'язки на систему автоматизованої перевірки

Тема 2: Логіка мови програмування. (8 годин)

Умовний оператор. Складений оператор. Складені умови. Тип Boolean.

Учні повинні знати:

- правила складання простих та складених умов;
- призначення та правила описування вказівок розгалуження;

Учні повинні вміти:

- складати та реалізовувати розгалуження;
- налагоджувати програми з розгалуженням;

Тема 3: **Організація циклів.** (8 годин)

Цикли. Види циклів та їх призначення. Вкладені цикли. Розв'язування задач на застосування циклів.

Учні повинні знати:

- призначення та правила користування вказівками повторення.

Учні повинні вміти:

- складати та реалізовувати циклічні програми з використанням циклів з постумовою, з передумовою та з параметром;
- налагоджувати програми з циклами.

Тема 4: Процедури. Формальні та фактичні параметри. (4 години)

Опис процедур. Процедури. Параметри-змінні.

Учні повинні знати:

сутність поняття процедур;
правила опису та використання процедур користувача;

Учні повинні вміти:

створювати та застосовувати процедури;
налагоджувати програми з процедурами.

Тема 5: **Функції.** (4 години)

Опис функцій. Звернення до функції.

Учні повинні знати:

сутність поняття функції;
правила опису та використання функцій користувача;

Учні повинні вміти:

створювати та застосовувати функції;
налагоджувати програми з функціями.

Тема 6: **Масиви.** (12 годин)

Призначення масивів. Створення одномірних та двомірних масивів. Організація масивів мовою програмування. *Пошук максимального та мінімального елемента масиву. Обробка масивів. Сортування масивів.*

Учні повинні знати:

- принципи побудови, опису масивів мовою програмування;
- призначення масивів;
- операції над масивами та елементами масивів;
- правила пошуку у масивах;
- правила обробки масивів;
- алгоритм пошуку мінімуму та максимуму у масиві;
- алгоритми сортування масивів;

Учні повинні вміти:

- застосовувати масиви;
- використовувати масиви при побудові програми;
- налагоджувати програми з масивами;
- використовувати вказівки масивів при побудові програм на пошук мінімуму ті максимуму серед множини елементів;
- налагоджувати програми з алгоритмами сортування;

Тема 7: **Масиви символів, рядкові величини.** (8 годин)

Тип даних рядок символів. Операції над рядками. Довжина рядка. Операція конкатенації.

Учні повинні знати:

- сутність поняття рядкового типу
- опис рядків у мові програмування;
- набір функцій та операцій для роботи з рядками програмування;
- сутність понятті довжини рядка;
- алгоритми для обробки рядків;
- сутність поняття конкатенації;

Учні повинні вміти:

- описувати та використовувати рядковий тип даних при складанні програм;
- налагоджувати програми з рядками;
- виконувати операції над рядками;
- знаходити довжину рядка;
- виконувати програмно операцію конкатенації рядків.

Тема 8: **Рекурсивні функції та процедури.** (4 години)

Поняття про рекурентні формули та рекурентні алгоритми. Програмування рекурсії. Обчислення формул з факторіалами та прогресій за допомогою рекурентних процедур. Рекурсивні алгоритми та рекурсивні визначення. Рекурсивні процедури та функції. Рекурсія зсередини. Поняття обмінного швидкого сортування.

Учні повинні знати:

сутність поняття рекурсії;
призначення рекурсії;
принцип побудови рекурентних алгоритмів;

Учні повинні вміти:

застосовувати рекурентні формули;
програмно обчислювати факторіали та прогресії;
складати та реалізовувати програми на рекурсію;
налагоджувати програми з рекурсіями.

Тема 9: **Використання множин.** (4 години)

Значення типу множина. Відношення і операції на множинах. Внутрішнє представлення множин.

Учні повинні знати:

сутність поняття типу множина;
допустимі операції на множинах.

Учні повинні вміти:

описувати та використовувати тип множин при складанні програм;

Тема 10: **Робота з файлами даних.** (4 години)

Організація та обробка файлів. Файлова змінна. Види файлів – текстовий, типізований, не типізований, файл типу запис. Пов'язання файлу з файловою змінною. Відкриття файлу для читання, запису, доповнення. Операції читання з файлу та запису у файл. Закриття файлу.

Учні повинні знати:

призначення файлів у *мові* програмування;
типи файлів у мові програмування;
опис файлової змінної та її призначення;
сутність пов'язання файлу та файлової змінної;
способи відкриття файлів;
правила описання вказівок відкриття файлу мовою програмування;
правила читання та запис інформації у файл;
правила закриття файлу.

Учні повинні вміти:

працювати з файлами у середовищі програмування;
складати та реалізовувати програми для роботи з файлами;
налагоджувати програми з файлами;
використовувати різні типи файлових змінних;
організовувати запис та читання інформації у файл.

Тема 11: **Структури даних. Записи.** (4 години)

Організація та обробка структури типу запис. Робота з записами.

Учні повинні знати:

призначення та правила опису структури запис;

Учні повинні вміти:

організовувати програмно тип запис та застосовувати його;
налагоджувати програми з використанням типу запис.

Календарне планування

№	Дата	Тема уроку	Примітка
1		Основні поняття мови програмування	
2		Основні поняття мови програмування	
3		Контроль через контестер	
4		Аналіз та корекція	
5		Основні поняття мови програмування	
6		Основні поняття мови програмування	
7		Контроль через контестер	
8		Аналіз та корекція	
9		Логіка мови програмування	
10		Логіка мови програмування	
11		Контроль через контестер	
12		Аналіз та корекція	
13		Логіка мови програмування	
14		Логіка мови програмування	
15		Контроль через контестер	
16		Аналіз та корекція	
17		Організація циклів	
18		Організація циклів	
19		Контроль через контестер	
20		Аналіз та корекція	
21		Організація циклів	
22		Організація циклів	
23		Контроль через контестер	
24		Аналіз та корекція	
25		Процедури. Формальні та фактичні параметри	
26		Процедури. Формальні та фактичні параметри	
27		Контроль через контестер	
28		Аналіз та корекція	
29		Функції	
30		Функції	
31		Контроль через контестер	
32		Аналіз та корекція	
33		Масиви	
34		Масиви	
35		Контроль через контестер	
36		Аналіз та корекція	
37		Масиви	
38		Масиви	
39		Контроль через контестер	
40		Аналіз та корекція	
41		Масиви	
42		Масиви	
43		Контроль через контестер	

44	Аналіз та корекція
45	Масиви символів, рядкові величини
46	Масиви символів, рядкові величини
47	Контроль через контестер
48	Аналіз та корекція
49	Масиви символів, рядкові величини
50	Масиви символів, рядкові величини
51	Контроль через контестер
52	Аналіз та корекція
53	Рекурсивні функції та процедури
54	Рекурсивні функції та процедури
55	Контроль через контестер
56	Аналіз та корекція
57	Використання множин
58	Використання множин
59	Контроль через контестер
60	Аналіз та корекція
61	Робота з файлами даних
62	Робота з файлами даних
63	Контроль через контестер
64	Аналіз та корекція
65	Структури даних. Записи
66	Структури даних. Записи
67	Контроль через контестер
68	Аналіз та корекція. Підсумок роботи факультативу

Вступ.
На цій мові написані найпопулярніші на сьогоднішній день операційні системи: Windows Linux. Мови програмування С++ може бути застосована до вирішення практично будь-яких задач.

Робота у середовищі.

Розділ І. Основні поняття мови програмування.

Мова програмування, як і будь-яка інша мова, містить чотири основних елементи: **символи, слова, словосполучення, речення**. **Слова**, що мають самостійний зміст, називають *лексемами*. **Словосполучення**, що задають правило обчислення деякого значення, називають *виразом*. **Речення**, які задають закінчений опис, називають *операторами*. Оператори можуть бути виконуваними та не виконуваними. Виконуваний оператор задає дію над даними. Не виконуваний оператор слугує для опису даних. **Програма** це об'єднана єдиним алгоритмом сукупність операторів та описів. Програми пишуть для того, щоб обробляти дані. Дані різних типів по-різному зберігаються у пам'яті комп'ютера. Від типу даних залежить:

- внутрішнє подання даних у пам'яті комп'ютера;
- множина значень, яких можуть набувати величини цього типу;
- операції та функції, які можна застосовувати до величин цього типу.

Обов'язків опис типу дозволяє компілятору робити перевірку допустимості різних конструкцій програми.

Стандартні типи даних

Основні типи даних часто називають арифметичними, оскільки їх можна використовувати у арифметичних операціях.

Стандартні типи даних С++

Назва	Позначення	Діапазон значень	Розмір, байт
Байт	`char`	від-128 до+127	1
Байт без знака	`unsigned char`	від 0 до 255	1
Ціле число	`int`	від -2147483648 до + 2147483647	4
Логічне значення	`bool`	значення true (істина) або false (хибно)	1
Дійсне число одинарної точності	`float`	від ±3.4e-38 до ± 3.4e+38 (7 значущих цифр)	4
Дійсне число подвійної точності	`double`	від ± 1.7e-308 до ±1.7e+308 (15 значущих цифр)	8
Дійсне число збільшеної точності	`long double`	від ±1.2e-4932 до ±1.2e+4932	10

Стандартні функції

Компіляція і виконання програми

Програма на мові С++ – це текст. За допомогою довільного текстового редактора програміст записує інструкцію, відповідно до якої комп'ютер працюватиме, виконуючи дану програму. Для того, щоб комп'ютер міг виконати програму, написану на мові С++, її потрібно перекласти мовою машинних інструкцій. Цю задачу вирішує компілятор. Компілятор читає файл з текстом програми, аналізує її, перевіряє на предмет можливих помилок і, якщо таких не виявлено, створює виконуваний файл, тобто файл з машинними інструкціями, який можна виконувати. Відкомпілювавши програму один раз, її можна виконувати багато разів, з різними вихідними даними. Приклади файлів:

`z1.cpp` – файл з текстом програми на мові С++; `z1.exe` – виконуваний файл

Абетка (алфавіт) та ключові слова.

Алфавіт мови С++ складається з:

- великих і малих літер латинського алфавіту: `"A", ..., "Z", "a", ..., "z";`
- цифр `0, 1, ..., 9;`
- спеціальних символів: `" '()[] {} < > .,;:?!~*+- = / \ | # % $ & ^ @` та символу підкреслення `_`.

Програми складаються із синтаксичних конструкцій, які називаються *оператори* (інші назви - командами, вказівки, речення). Команди будуються з *лексем* - неподільних елементів мови: слів, чисел, символів операцій.

Слова поділяють на ключові слова й ідентифікатори.

Ідентифікатори.

Ідентифікатор - це назва (ім'я), яку користувач надає об'єктам, наприклад, змінним, сталим, функціям. Ідентифікатори в мові С++ – це послідовність знаків, що починається з букви або знаку підкреслення. У ідентифікаторах можна використовувати заголовні і рядкові латинські букви, цифри і знак підкреслення. Довжина ідентифікаторів довільна. Приклади правильних ідентифікаторів:

```
abc A12 NameOfPerson    BYTES_PER_WORD
```

Відзначимо, що `abc` і `Abc` – два різні ідентифікатори, тобто заголовні і рядкові букви розрізняються. Приклади неправильних ідентифікаторів:

```
12X     a-b
```

Ряд слів в мові С++ мають особливе значення і не можуть використовуватися як ідентифікатори. Такі зарезервовані слова називаються ключовими. Список ключових слів:

```
asm          auto             bad_cast
bad_typeid   bool             break
case         catch            char
class        const            const_cast
continue     default          delete
do           double           dynamic_cast
else         enum             extern
float        for              friend
goto         if               inline
int          long             mutable
namespace    new              operator
private      protected        public
register     reinterpret_cast return
short        signed           sizeof
static       static_cast      struct
switch       template         then
this         throw            try
type_info    typedef          typeid
union        unsigned         using
virtual      void             volatile
while        xalloc
```

Змінити призначення ключового слова у програмі не можна.

Змінні.

Програма оперує інформацією, представленою у вигляді різних об'єктів і величин.

Змінна – це символічне позначення величини в програмі. Як ясно з назви, значення змінної (або величина, яку вона позначає) під час виконання програми може змінюватися. З точки зору архітектури комп'ютера, змінна – це символічне позначення елементу оперативної пам'яті програми, в якій зберігаються дані. Вміст цієї чарунки – це поточне значення змінної. У мові С++ перш ніж використовувати змінну, її необхідно оголосити. Оголосити змінну з ім'ям `x` можна так:

```
int x;
```

У оголошенні першим вказується назва типу змінною `int` (ціле число), а потім ідентифікатор `x` – ім'я змінної. В змінної `x` є тип – в даному випадку ціле число. Тип змінної

визначає, яких можливих значень ця змінна може набувати і які операції можна виконувати над даною змінною. Типа змінної змінити не можна, тобто доки змінна `x` існує, вона завжди буде цілого типа.
Мова С++ – строго типізована мова. Будь-яка величина, яка використовується у програмі, належить до якого-небудь типу. При будь-якому використанні змінних в програмі перевіряється, чи дозволена операція до типу змінної.
Відповідність типів перевіряється під час компіляції програми. Якщо компілятор виявляє невідповідність типу змінної і її використання, він видасть помилку (або попередження). Проте під час виконання програми відповідність типів не перевіряються. Такий підхід, з одного боку, дозволяє виявити і виправити велику кількість помилок на стадії компіляції, а, з іншого боку, не уповільнює виконання програми.
Змінній можна присвоїти яке-небудь значення за допомогою операції присвоєння. ***Присвоїти*** – означає встановити поточне значення змінної. По-іншому можна пояснити, що операція присвоєння запам'ятовує нове значення в чарунці пам'яті, яка позначена змінною.

```
int x;       // оголосити цілу змінну x
int y;       // оголосити цілу змінну y
x = 0;       // присвоїти x значення 0
y = x + 1;   // присвоїти y значення x + 1,тобто 1
x =x + 1;       // x значення змінної x збільшили 1
y = x + 1;   // присвоїти y значення x + 1, тобто уже 2
```

У математиці вираз `x = x + 1;` не має сенсу, а в мові С++ означає, що значення змінної з чарунки **`x`** збільшили на **`1`** і нове значення збережено у цій же чарунці.

Константи.
В програмі можна явно записати величину – число, символ і тому подібне. Наприклад, ми можемо записати вираз `x + 4` – додати поточне значення змінної **`x`** і число **`4`**. Значення змінної x може бути різним. Проте ціле число чотири завжди залишиться тим самим. Це незмінна величина або константа. Отже, явний запис значення в програмі – це константа. Далеко не завжди зручно записувати константи в тексті програми явно. Набагато частіше використовуються символічні константи. Наприклад, якщо ми запишемо

```
const int BITS_IN_WORD = 32;
```

то потім ім'я `BITS_IN_WORD` можна буде використовувати замість цілого числа **`32`**.
Дані, значення яких не змінюються в процесі виконання програми, називають константами.
Модифікатор **`const`** показує, що дане значення змінювати не можна.

Вирази.
Програма оперує з даними. Числа можна складати, віднімати, умножати, ділити. З різних величин можна складати вирази, результат обчислення яких – нова величина. Наведемо приклади виразів:

```
X * 12 + Y  // значення X помножити на 12 і до результату додати значення Y
val < 7     // порівняти значення val з 7
-9          // константний вираз -9
```

Вираз, після якого стоїть крапка з комою, – це оператор-вираз. Його сенс полягає в тому, що комп'ютер повинен виконати всі дії, записані в даному виразі.

```
x + y - 12; // додати значення x та y а потім відняти 12
a = b + 1; //додати одиницю до значення b і запам'ятати результат в змінній a
```

Вирази – це змінні, функції і константи (операнди), об'єднані знаками операцій. Операції можуть бути унарними – з одним операндом, наприклад, мінус; можуть бути бінарні – з двома операндами, наприклад складання або ділення. У С++ є навіть одна операція з трьома операндами – умовний вираз.

Операція присвоювання.
Присвоювання – це теж операція, вона є частиною виразу. Значення правої частини виразу (операнда) присвоюється лівій частині виразу (операнду).

```
x = 2;          // змінній x присвоїли значення 2
cond = x < 2; // змінна cond набуде значення true, якщо x менше 2,
                // інакше набуде  значення false
3 = 5;  // помилка, число 3 не може змінювати своє значення
```

Розглянемо фрагмент програми:

```
int x = 0;
x = 3;
x = 4;
x = x + 1;
```

Спочатку оголошується змінна x з початковим значенням `0`. Після цього значення x змінюється на 3, 4 і потім 5. Зверніть увагу на останній рядок. При виконанні операції присвоювання спочатку обчислюється правий операнд (права частина виразу), а потім лівий. Коли обчислюється вираз x + 1, значення змінної x дорівнює 4. Тому значення вираження x + 1 рівне 5. Після виконання операції присвоювання значення змінної x стає рівним 5.

В операції присвоювання теж є результат. Він дорівнює значенню лівого операнда. Таким чином, операція присвоювання може брати участь в складнішому виразі:

```
z = (x = y + 3);
```

У наведеному прикладі змінним `x` та `z` присвоюється значення `y + 3`.

Дуже часто в програмі доводиться значення змінної збільшувати або зменшувати на одиницю. Для того, щоб зробити ці дії найбільш ефективними і зручними для використання, застосовуються передбачені в C++ спеціальні знаки операцій: ++ (збільшити на одиницю) і -- (зменшити на одиницю). Існує дві форми цих операцій: префіксна і постфіксна. Розглянемо їх на прикладах.

```
int x = 0;
++x;
```

Значення `x` збільшується на одиницю і стає рівним 1.

```
--x;
```

Значення `x` зменшується на одиницю і стає рівним 0.

```
int y = ++x;
```

Значення `x` знову збільшується на одиницю. Результат операції ++ нове значення `x`, тобто змінній `y` присвоюється 1.

```
int z = x++;
```

Тут використовується постфіксний запис операції збільшення на одиницю. Значення змінної `x` до виконання операції дорівнює 1. Сама операція та ж – значення `x` збільшується на одиницю і стає рівним 2. Проте результат постфіксної операції – це значення аргументу до збільшення. Таким чином, змінною z присвоюється значення 1. Аналогічно, результатом постфіксної операції зменшення на одиницю є початкове значення операнда, а префіксною – його кінцеве значення.

Досить часто одна і та ж змінна використовується в лівій і правій частині операції присвоєння, наприклад:

```
x = x + 5;
y = y * 3;
z = z – (x + y);
```

У C++ ці вирази можна записати лаконічніше:

```
x += 5;
y *= 3;
z -= x + y;
```

Окрім стислості виразу, такий запис полегшує оптимізацію програми компілятором.

Всі операції мови C++

Поряд із загальноприйнятими арифметичними і логічними операціями, в мові C++ є набір операцій для роботи з бітами – порозрядні `I`, `АБО`, `ЩО ВИКЛЮЧАЄ АБО` і `НЕ`, а також зсув.

Операція **`sizeof`**. Ця операція дає змогу визначити, скільки пам'яті займає та чи інша змінна. Наприклад:

```
sizeof(long);  // скільки байтів займає тип long
sizeof (b);// скільки байтів займає змінна b
```

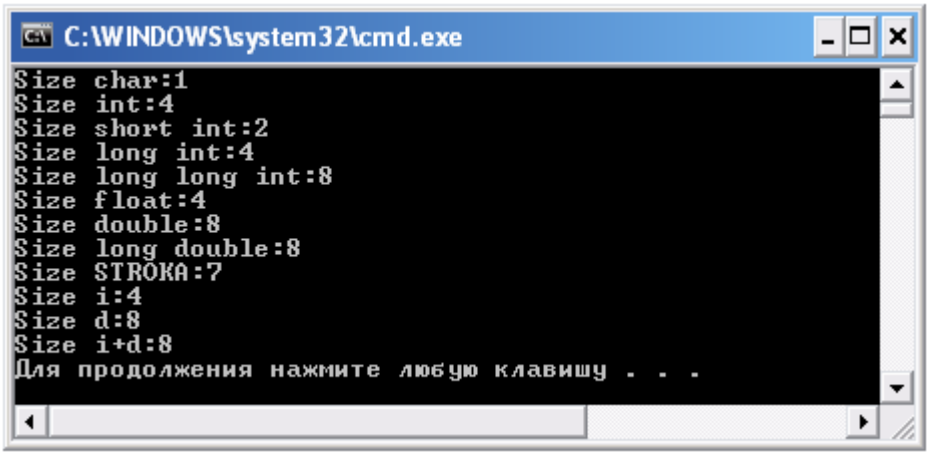
Операція **sizeof** як аргумент бере ім'я типу або виразу. Аргумент поміщають в дужки (якщо аргумент – вираз, дужки не обов'язкові). Результат операції – ціле число, рівне кількості байтів, яка необхідна для зберігання в пам'яті заданої величини.

Приклад 1. наступна програма визначає розміри об’єктів і типів в байтах. Результат обчислення виведемо на екран.

```
#include"iostream"
using namespace std;
int main()
{
    int i=3;
      double d=0.2;
      cout<<"Size char:"<<sizeof(char)<<"\n";
      cout<<"Size int:"<<sizeof(int)<<"\n";
      cout<<"Size short int:"<<sizeof(short int)<<"\n";
      cout<<"Size long int:"<<sizeof(long int)<<"\n";
      cout<<"Size long long int:"<<sizeof(long long int)<<"\n";
      cout<<"Size float:"<<sizeof(float)<<"\n";
      cout<<"Size double:"<<sizeof(double)<<"\n";
      cout<<"Size long double:"<<sizeof(long double)<<"\n";
      cout<<"Size STROKA:"<<sizeof"STROKA"<<"\n";
      cout<<"Size i:"<<sizeof i<<"\n";
      cout<<"Size d:"<<sizeof d<<"\n";
      cout<<"Size i+d:"<<sizeof (i+d)<<"\n";

}
```

Результат виконання:



Завдання: В останньому рядку програми заберіть круглі дужки. Запустіть програму на виконання та прокоментуйте отриманий результат.

Арифметичні операції

```
+ додавання
- віднімання
* множення
/ ділення
```

Операції додавання, віднімання, множення і ділення цілих і дійсних чисел. Результат операції – число, за типом відповідне більшому по розрядності операнду. Наприклад, додавання чисел типа **short** і **long** в результаті дає число типа **long**

```
% залишок
```

Операція знаходження залишку від ділення одного цілого числа на інше. Тип результату – ціле число.

```
- мінус
+ плюс
```

Операція "мінус" – це унарна операція, при якій знак числа змінюється на протилежний. Вона застосовна до будь-яких чисел із знаком. Операція "плюс" існує для симетрії. Вона нічого не робить, тобто застосована до цілого числа, його ж і видає.

```
++ збільшення на 1, префіксна і  постфіксна форми
-- зменшення на 1, префіксна і   постфіксна форми
```

Ці операції інколи називають "автозбільшенням" (інкремент) і "автозменшенням" (декремент). Вони збільшують (або, відповідно, зменшують) операнд на одиницю. Різниця між постфіксною (знак операції записується після операнда, наприклад x++) і префіксною (знак операції записується перед операндом, наприклад --y) операціями полягає в тому, що в першому випадку результатом є значення операнда до зміни на одиницю, а в другому випадку – після зміни на одиницю.

Операції порівняння

```
== рівно
!= не рівно
< менше
> більше
<= менше або рівно
>= більше або рівно
```

Операції порівняння. Порівнювати можна операнди будь-якого типа, але вони мають бути обидва одного і того ж вбудованого типу (порівняння на рівність і нерівність працює для двох величин будь-якого типу), або між ними має бути визначена відповідна операція порівняння. Результат – логічне значення **`true`** або **`false`**.

Логічні операції.

```
&& логічне і
|| логічне або
! логічне НЕ
```

Логічні операції: кон'юнкції, диз'юнкції і заперечення. Як операнди виступають логічні значення, результат – теж логічне значення **`true`** або **`false`**.

Бітові операції.

```
& бітове і
| бітове АБО
^ бітове ВИКЛЮЧЕННЯ АБО
~ бітове НЕ
```

Побітові операції над цілими числами.

Відповідна операція виконується над кожним бітом операндів. Результатом є ціле число.

```
 <<     зсув вліво
 >>     зсув вправо
```

Побітовий зсув лівого операнда на кількість розрядів, відповідну значенню правого операнда. Результатом є ціле число.

Умовна операція

```
операнд1?операнд2:операнд3
```

Тернарна операція; якщо значення першого операнда – істина, то результат – другий операнд; якщо хибне – результат – третій операнд. Перший операнд має бути логічним значенням, другий і третій операнди можуть бути будь-якого, але одного і того ж, типу, а результат буде того ж типу, що і третій операнд. Наприклад:

```
y=x<0?-x:x;
```

Прокоментуйте цей вираз самотійно.

Послідовність.

```
, послідовність
```

Виконати вираз до коми, потім вираз після коми. Два довільні вирази можна поставити поруч, розділивши їх комами. Вони виконуватимуться послідовно, і результатом всього виразу буде результат останнього.

Операції присвоювання.

```
= присвоєння
```

Присвоїти значення правого операнда лівому. Результат операції присвоєння – це значення правого операнда.

```
+=, -=, *=, /=, %=, |=, &=, ^=, <<=, >>=
виконати операцію і присвоїти
```

Виконати відповідну операцію з лівим операндом і правим операндом і присвоїти результат лівому операнду. Типи операндів мають бути такими, що, по-перше, для них має бути визначена відповідна арифметична операція, а по-друге, результат може бути присвоєний лівому операнду.

Операція перетворення типів.

Для приведення виразу до іншого типу даних у С++ існує операція перетворення типів:

```
(тип) вираз;
```

Наприклад, в результаті дій: `float z; int y; int x=5; y=x/2; z=(float)x/2;` змінна `y` прийме значення 2, а змінна `z` значення 2.5.

Завдання: яке значення набуде зміна z після виконання наступних дій:

```
float z; int x=5; z=x/2;
```

Порядок обчислення виразів

У кожної операції є пріоритет. Якщо у виразі декілька операцій, то першою буде виконана операція з вищим пріоритетом. Якщо ж операції одного і того ж пріоритету, вони виконуються зліва направо. Наприклад, у виразі

```
2 + 3 * 6
```

спочатку буде виконано множення, а потім додавання і значення цього виразу рівне 20.

У виразі

```
2 * 3 + 4 * 5
```

спочатку буде виконано множення, а потім додавання. У якому порядку буде виконуватися множення не відомо чи спочатку 2 * 3, а потім 4 * 5 чи навпаки. Таким чином для операції додавання порядок обчислення її операндів не визначений.

У виразі

```
x = y + 3
```

спочатку виконується додавання, а потім присвоєння, оскільки пріоритет операції присвоєння нижчий, ніж пріоритет операції додавання. Для даного правила існує виключення: якщо у виразі декілька операцій присвоєння, то вони виконуються справа наліво. Наприклад, у виразі

```
x = y = 2
```

спочатку буде виконана операція надання змінній `y` значення 2. Потім результат цієї операції – значення 2 – присвоюється змінній `x`.

Для того, щоб змінити послідовність обчислення виразів, можна скористатися круглими дужками. Частина виразу,яка поміщена у дужки, обчислюється в першу чергу. Значенням виразу `(2 + 3) * 6` буде 30.

Приклад 1.1

Задані дві цілочисельні змінні a та b. Поміняти їх значення місцями.

```
#include<iostream>
using namespace std;
int main()
   {
         int a=5,b=8,c;
         c=a; a=b; b=c;
         cout<<a<<' '<<b;
   }
```

Приклад 1.2

Задані дві цілочисельні змінні a та b. Поміняти їх значення місцями,без використання допоміжної змінної.

```
#include<iostream>
using namespace std;
int main()
   {
         int a=5,b=8,c;
         c=a; a=b; b=c;
         cout<<a<<' '<<b;
```

```
}
```

Приклад 1.3

Задані дві цілочисельні змінні **a** та **b.** Поміняти їх значення місцями, використавши логічну операцію ^(виключення або) .

Вхідні дані: a b

Вихідні дані: b a

```
#include<iostream>
using namespace std;
int main()
   {
         int a=5,b=8;
         a=a^b; b=a^b; a=a^b;
         cout<<a<<' '<<b;
    }
```

Оператори.

Запис дій, які повинен виконати комп'ютер, складається з операторів. При виконанні програми оператори виконуються один за одним, якщо лише оператор не є оператором управління, який може змінити послідовне виконання програми. Розрізняють оператори оголошення імен, оператори управління і оператори-вирази.

Оператори-вирази.

Вираз, який закінчується крапка з комою, – це оператор-вираз. Його сенс полягає в тому, що комп'ютер повинен виконати всі дії, записані в даному виразі, іншими словами, обчислити значення виразу. Найчастіше у таких операторах є операція присвоєння або виклик функції. Оператори виконуються послідовно, і всі зміни значень змінних, зроблені в попередньому операторові, використовуються в наступних.

```
a  =  1;
b  =  3;
m  =  max(a,b);
```

Оператори оголошення імен

Ці оператори оголошують імена, тобто роблять їх відомими програмі. Всі ідентифікатори або імена, використовувані в програмі на мові С++, мають бути оголошені. Оператор оголошення складається з назви типу і оголошуваного імені:

```
int   x;          // оголосити цілу змінну x
double f;         // оголосити  змінну f типу double
const float pi = 3.1415;//оголосити константу pi типу float із значенням 3.1415
```

Оператор оголошення закінчується символом крапка з комою.

Оператор введення

Ввід - вивід даних у мові С++ здійснюється або з допомогою функцій вводу –виводу у стилі мови програмування С, або з використанням бібліотеки класів С++. Перевага об’єктів вводу – виводу С++ у тому, що вони простіші у використанні, особливо коли ввід-вивід достатньо простий. Функції вводу-виводу, успадковані від мови програмування С, є громіздкі, але підходять до задач із складним форматним вводом-виводом.

Функції вводу - виводу.

Функція :

```
printf(рядок форматів, список змінних);
```

виконує форматний вивід змінних, які вказані у списку змінних, у відповідності до рядку форматів.

Функція :

```
scanf(рядок форматів, список змінних);
```

виконує форматний ввід змінних, адреси яких вказані у списку змінних, у відповідності до рядку форматів.
Рядок форматів містить символи, які будуть виводитися на екран або ввід буде здійснюватися із клавіатури та так звані специфікації. Специфікації – це рядки, які починаються символом % і виконують керування форматом:

```
%мітка ширина.точність модифікатор тип
```

Параметри `мітка ширина,точність,модифікатор` у специфікаціях можуть бути відсутніми. Значення параметрів специфікації подано у таблиці.
Таблиця Символи керування.

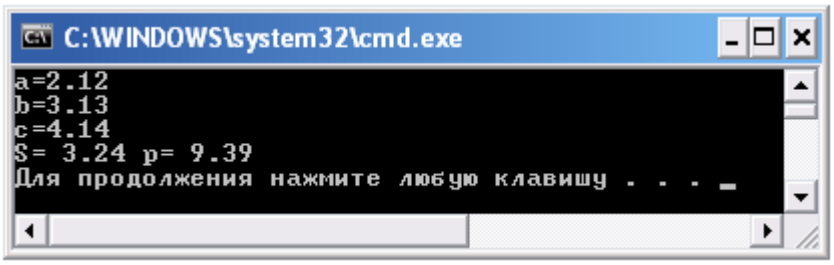
Мітка	Призначення
	точність
нічого	Точність по замовчуванню
n	Для типів **e, f, E,** виводить знаків **n** після коми
	модифікатор
h	Для **d, i, o, u, x, X** тип **short int**
l	Для **d, i, o, u, x, X** тип **long int**
	тип
c	При вводі символьний тип **char** , при виводі один байт
d	Десяткове число типу **int** із знаком
i	Десяткове число типу **int** із знаком
o	Число у системі числення 8
u	Десяткове число типу **int unsigned**
x, X	Число у системі числення 16
f	Число із знаком виду **[-] dddd.dddd**
e	Число із знаком виду **[-] d.ddddе[+/-]ddd**
E	Число із знаком виду **[-] d.ddddE[+/-]ddd**
g	Число із знаком типу **e** або **f** в залежності від значення і точності
G	Число із знаком типу **E** або **f** в залежності від значення і точності
s	Рядок символів

Крім того рядок форматів може містити деякі спеціальні символи, які подані у таблиці:
Таблиця 1.2 Спеціальні символи.

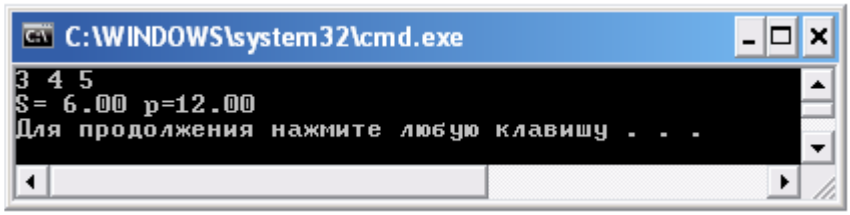
Символ	Призначення
\b	Зсув поточної позиції вліво
\n	Перевід рядка
\'	Символ одинарних лапок
\"	Символ подвійних лапок
\?	Символ ?
\r	Перевід на початок рядка, не переходячи на новий рядок
\t	Горизонтальна табуляція

Першим рядком програми, у якій будуть застосовані функції вводу-виводу у стилі С, повинна бути директива `#include<stdio.h>`. Заголовковий файл `stdio.h` містить опис функцій вводу – виводу.
Розглянемо роботу функцій на прикладі такої задачі:
Приклад 1.4:
Відомі сторони трикутника ***a,b,c***. Обчислити його периметр ***p*** та площу ***S***.
Вхідні дані: a b c
Вихідні дані подати у вигляді: S= p=
Для обчислення площі застосуємо формулу Герона: $S = \sqrt{r \cdot (r-a) \cdot (r-b) \cdot (r-c)}$, де $r = \frac{a+b+c}{2}$ - півпериметр.

```
#include "stdafx.h" // Приклад 1.4 варіант перший
#include<stdio.h>
#include<math.h>
using namespace System;
int main()
{
    float a,b,c,S,r;
      printf("a=");
      scanf("%f",&a);
      printf("b=");
      scanf("%f",&b);
      printf("c=");
      scanf("%f",&c);
      r=(a+b+c)/2;
      S=sqrt(r*(r-a)*(r-b)*(r-c));
      printf("S=%5.2f\t",S);
      printf("p=%5.2f\n",2*r);
    return 0;
}
```




```
#include "stdafx.h" // Приклад 1.4 варіант другий
#include<stdio.h>
#include<math.h>
using namespace System;
int main()
{
    float a,b,c,S,r;
      scanf("%f%f%f",&a,&b,&c);
      r=(a+b+c)/2;
      S=sqrt(r*(r-a)*(r-b)*(r-c));
      printf("S=%5.2f\tp=%5.2f\n",S,2*r);
      return 0;
}
```



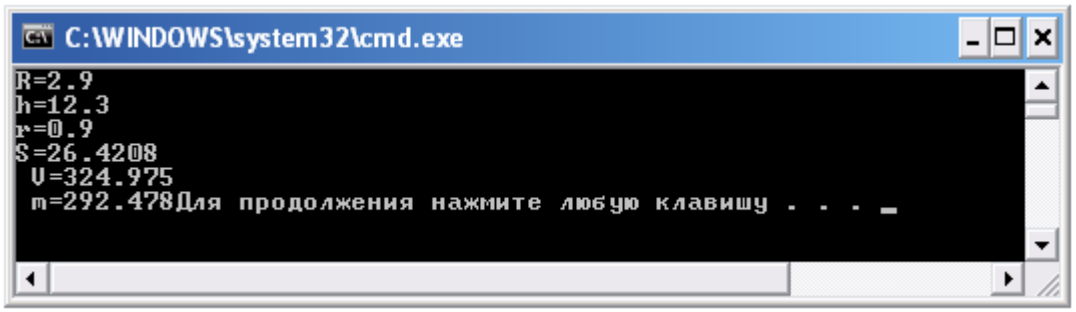
Об'єктно орієнтовані засоби вводу виводу.

Приклад 1.5
Металеве тіло має форму циліндра з радіусом основи ***R*** та висотою ***h***. Визначити об'єм ***V***, масу тіла ***m*** та площу основи циліндра ***S***, якщо відомо, що ***ρ*** - густина металу.
Вхідні дані: *ρ h R*
Вихідні дані: S V m

Врахуємо що площу основи циліндра можна обчислити за формулою $S = 2 \cdot \pi \cdot R$, об'єм $V = \pi R^2 h$, масу $m = \rho \cdot V$

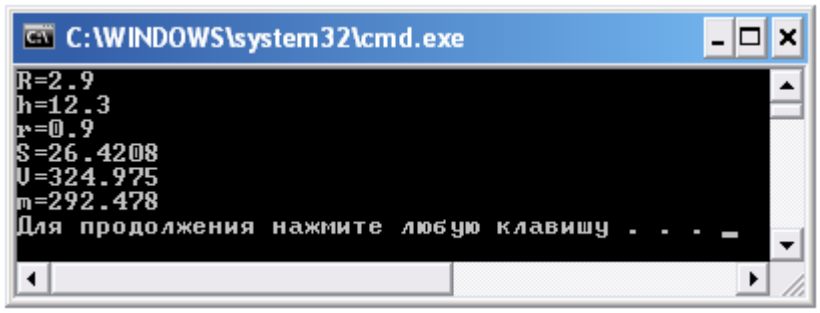
```
#include "stdafx.h" // Приклад 1.5
#include<iostream>
using namespace std;
#define pi 3.14159  // визначення константи
int main()
{
    double R,h,S,r,V,m;  // опис змінних
      cout<<"R="; // вивід на екран символів R=
      cin>>R; // ввід з клавіатури значення R
      cout<<"h=";
      cin>>h;
      cout<<"r=";
      cin>>r;
      S=pi*R*R;  // обчислення площі
      V=S*h;// обчислення обєму
      m=V*r;// обчислення маси
      cout<<"S="<<S; // вивід на екран символів S= та значення площі
      cout<<"\n V="<<V;
      cout<<"\n m="<<m;
      return 0;
}
```

Результат виконання програми



Творче завдання.

Внесіть зміни у програму, щоб картинка на екрані була така:



Структура програм

Директиви препроцесора.

Препроцесор - це програма, яка опрацьовує директиви. ***Директиви препроцесора*** - це команди компілятора, які виконуються на початку компіляції програми. Директиви мови С++ починаються із символу **#**. Розглянемо декілька типів директив.

Директива **`include`** означає, що до програми необхідно Приєднати програмний код із зазначеного після неї файлу.

Файли, які приєднують директивою **`#include`** (читати: *паунд інклуд*), називаються ***файлами заголовків*** (*header*- файлами, бібліотеками, модулями). У таких файлах зазвичай оголошують сталі й змінні, заголовки (сигнатури) функцій тощо.

Усі стандартні команди та функції мови С++ визначенні у файлах заголовків. Щоб приєднати модуль до програми користувача, директиву препроцесора необхідно зазначити на початку програми так:

```
#include <назва файлу.розширення>
```

або так:

```
#include "шлях до файлу\назва файлу.розширення"
```

Зазвичай усі стандартні бібліотеки розміщені у папці `INCLUDE` середовища С++. У такому випадку назва файлу є параметром директиви, її зазначають у кутових дужках `<назва>`, наприклад
`#include <math.h>`.
Якщо ж потрібний файл розміщений не у папці `INCLUDE`, то назву файлу із зазначенням шляху пишуть у лапках "...".

Директива **`#define`** має подвійне значення. По-перше, вона може задати стале значення (оголошує сталу). Наприклад, якщо у програмі задано **`#define`** `N 25`, то N під час виконання програми матиме значення 25. По-друге, вона дає змогу описати *макроси* - короткі команди (пере означити команди) чи записати функції, наприклад, так:

```
#define min(a,b) (a<b)?a:b;
```

Тепер скрізь для знаходження мінімуму із двох чисел можна писати так:

```
d=min(a,b);
```

Директива **`#undef`** скасовує дію директиви **`#define`**.

Завдання для самоконтролю
Якщо в умові задачі нічого не сказано про тип змінних, то ці змінні можуть бути як цілі, так і дійсні. Отже описувати їх слід REAL. Після кожної задачі буде міститися контрольний приклад вхідних та вихідних даних. Це означає те, що коли ви введете при запуску програми вхідні дані, то ваша програма має вивести результат, який співпаде із вихідними даними. Увага! Правильний вивід вихідних даних не гарантує того, що ваша програма буде правильно працювати при інших вхідних даних. Перевірку треба обов'язково продовжити самостійно.

Задача 1-1.
Дано цілі числа a, b, що ж катетами прямокутного трикутника. Скласти програму знаходження його площі. В результаті вивести два знаки після коми.
Вхідні дані
4 5
Вихідні дані
10.00

Задача 1-2.
Дано чотири дійсні числа a, b, c, d. Знайти їх суму. В результаті вивести чотири знаки після коми.
Вхідні дані
3 4 2.5 1
Вихідні дані
10.5000

Задача 1-3.
Знайти периметр прямокутника, якщо відомо дві його сторони a та b. Сторони прямокутника цілі числа.
Вхідні дані
2 6
Вихідні дані
16

Задача 1-4.
Дано цілі числа x, y. В першому рядку вивести залишок від ділення першого числа на друге, а в другому залишок від ділення другого на перше.
Вхідні дані
15 16
Вихідні дані
15
1

Задача 1-5.
Дано дійсне число p. В першому рядку вивести його заокруглене ціле значення, в другому – його цілу частину, в третьому – його дробову частину (один знак після коми).
Вхідні дані.
3.5
Вихідні дані.
4
3
0.5

Задача 1-6

Дано цілі числа x та y. Знайти значення виразу: $x^3 + \frac{x+1}{y^2+1}$. В результаті вивести один знак після коми.
Вхідні дані.
3 1
Вихідні дані.
29.0

Задача 1-7
Дано цілі числа x, y. Знайти різницю першого та другого числа.
Вхідні дані.
10 4
Вихідні дані.
6

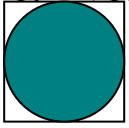
Задача 1-8
Знайти значення виразу: $\frac{2x^2+1}{x^2+1} + 5$. В результаті вивести два знаки після коми.

Вхідні дані.
0
Вихідні дані.
6.00

Задача 1-9
Дано дійсне число N. Вивести подвоєну його цілу частину.
Вхідні дані.
12.958
Вихідні дані.
24

Задача 1-10
Дано дійсні числа P і K. Дробову частину P помножити на K і вивести результат заокруглений до найближчого цілого.
Вхідні дані.
12.1257 1000
Вихідні дані.
126

Задача 1-11
На квадрат наклали круг такої величини, що він одночасно торкається кожної сторони квадрата. Сторона квадрата є ціле число a. Знайти найменше ціле число, що є більшим від сумарної площі частин квадрата видимих з під круга (круг не є прозорим).
Вхідні дані.
20
Вихідні дані.
86



Задача 1-12
Дано двоцифрове ціле число, що не закінчується нулем. Вивести частку від ділення першої цифри на другу.

Вхідні дані.
72
Вихідні дані.
3

Задача1-13
Дано натуральне трьохцифрове число. Вивести суму його цифр.
Вхідні дані.
123
Вихідні дані.
6

Задача 1-14
Координати точки цілі числа X і Y, причому X>0 та Y>0. Вивести у першому рядку координати точки симетричної даній відносно осі OX, у другому – відносно OY, у третьому – відносно початку координат і у четвертому площу фігури з вершинами у цих точках.
Вхідні дані.
1 2
Вихідні дані.
1 -2
-1 2
-1 -2
8

Задача 1-15
Дано два числа. Знайти суму їх дробових частин. В результаті вивести два знаки після коми.
Вхідні дані.
2.5 3.7
Вихідні дані.
1.20

Розділ 2. Логіка С++

Складений оператор.

Складений оператор – це група операторів, які відокремлені один від одного крапка з комою та поміщені у фігурні дужки {}:

```
{
        оператор_1;
        оператор_2;
        .
        .
        .
        оператор_n;

}
```

Транслятор сприймає складений оператор, як одне ціле.

Оператори управління.

Оператори управління визначають, в якій послідовності виконується програма. Якби їх не було, оператори програми завжди виконувалися б послідовно, в тому порядку, в якому вони записані. Оператори управління бувають: ***умовні та безумовні***.

Умовні оператори

Умовні оператори дозволяють вибрати один з варіантів виконання дій залежно від умов. Умова – це логічний вираз, тобто вираз, результатом якого є логічне значення **`true`** (істина) або **`false`** (хибно). Оператор `if` вибирає один з двох варіантів послідовності обчислень.

```
if   (умова) оператор1  else оператор2
```

Якщо умова істина, виконується оператор1, якщо хибна, то виконується оператор2.

```
if   (x > y) a = x; else a = y;
```

У даному прикладі змінній `a` присвоюється значення максимуму з двох величин `x` і `y`.
Конструкція `else` необов'язкова.
Якщо в разі істинності умови необхідно виконати декілька операторів, їх можна помістити у фігурні дужки:

```
if    (x < 0)
       {   x = -x;
              cout << "Змінити значення x на протилежне по знаку";
       }
```

Умовний оператор можна розширити для перевірки декількох умов:

```
if    (x  < 0)
       cout  << "Від'ємне значення";
else if    (x > 0)
       cout  << "Додатнє значення";
else
       cout  << "Нуль";
```

Конструкцій `else if` може бути декілька.

Оператор вибору

Хоча будь-які комбінації умов можна виразити за допомогою оператора `if`, досить часто запис стає незручним і заплутаним. Оператор вибору `switch` використовується, коли для кожного з декількох можливих значень виразу потрібно виконати певні дії. Наприклад, передбачимо, що в змінній **`code`** зберігається ціле число від 0 до 2, і нам потрібно виконати різні дії залежно від її значення:

```
switch (code) {
case 0:
     cout << "код нуль";
     x = x + 1;
     break;
case 1 :
```

```
            cout << "код один";
            y = y + 1;
            break;
      case 2:
            cout << "код два";
            z = z + 1;
            break;
      default:
            cout << "Неправильне значення";
      }
```

Залежно від значення **code** управління передається на одну з міток `case`. Виконання оператора закінчується після досягнення або оператора `break`, або кінця оператора `switch`. Таким чином, якщо **code** дорівнює 1, виводиться "код один", а потім змінна в збільшується на одиницю. Якби після цього не стояв оператор `break`, то управління "провалилося" б далі, була б виведена фраза "код два", і змінна z теж збільшилася б на одиницю. Якщо значення перемикача не збігається ні з одним із значень міток `case`, то виконуються оператори, записані після мітки `default`. Мітка `default` може бути опущена, що еквівалентно запису:

```
      default:
            ;  // пустий оператор, який не виконує ніяких дій
```

Зрозуміло, що наведений приклад можна переписати за допомогою оператора `if`:

```
if   (code == 0) {
 cout << "код нуль";
       x = x + 1;
} else if (code == 1) {
       cout << "код один";
             y = y + 1;
} else if (code == 2) {
       cout << "код два";
        z = z + 1;
} else {
  cout << " Неправильне значення ";
  }
```

Мабуть, запис за допомогою оператора перемикання `switch` компактніший та зрозуміліший.

Приклад 1. По заданому номеру дня місяця вивести назву дня тижня, якщо місяць містить 31 день і перше число припадає на понеділок.

```
#include "stdafx.h"
#include"stdio.h"
int main()
{
    unsigned int D,R;
      printf("\n D="); scanf("%d",&D);
      R=D%7;
      switch(R)
      {
            case 1:printf("Monday \n");break;
            case 2:printf("Tuesday \n");break;
            case 3:printf("Wednesday \n");break;
            case 4:printf("Thursday \n");break;
            case 5:printf("Friday \n");break;
            case 6:printf("Saturday \n");break;
            case 0:printf("Sunday \n");break;
      return 0;
      }
}
```

Завдання: модифікуйте програму таким чином, щоб вона перевіряла вхідні дані (тобто щоб номер дня не був більший за 31).

Завдання для самоконтролю

Задача 2-1

Дано ціле число X. Вивести Yes, якщо воно парне, або No коли X є непарним.

Вхідні дані

56

Вихідні дані.

Yes

Задача 2-2.

Дано цілі числа a та b. Вивести подвоєне більше число.

Вхідні дані

4 6

Вихідні дані.

12

Задача 2-3

Дано ціле числа a. Якщо воно парне – вивести його частку від ділення на 2, інакше 0.

Вхідні дані

22

Вихідні дані.

11

Задача 2-4

Дано цілі числа a та b. Вивести їх в один рядок в порядку зростання, тобто, спочатку менше, а потім більше.

Вхідні дані

6 1

Вихідні дані.

1 6

Задача 2-5

Дано дійсне число p. Вивести «Yes», якщо воно ціле і «No» в іншому випадку.

Вхідні дані

2.6

Вихідні дані.

No

Задача 2-6

Дано цілі числа a, b, c, d. В одному рядку вивести спочатку найменше з них, а потім найбільше.

Вхідні дані

2 8 1 10

Вихідні дані.

1 10

Задача 2-7

Дано цілі координати точки x, y. Точка не лежить на осі координат. Вивести «Yes», якщо точка лежить у другій чверті або «No» в іншому випадку.

Вхідні дані

2 -8

Вихідні дані.

No

Задача 2-8

Дано цілі координати точки x, y. Точка не лежить на осі координат. Вивести номер чверті, якій належить дана точка.

Вхідні дані

2 -8

Вихідні дані.

4

Задача 2-9

Є додатні цілі x, y та z. Потрібно з'ясувати, чи існує трикутник зі сторонами x, y, z. Вивести «Yes» або «No».

Вхідні дані

1 4 5

Вихідні дані.

No

Задача 2-10

Дано цілі числа a, b, c, d. Вивести «Yes», якщо вони є впорядковані за зростанням, інакше вивести «No».

Вхідні дані

3 4 4 6

Вихідні дані.

No

Задача 2-11

Дано натуральне число N. Вивести «Yes», якщо воно є квадратом натурального числа, інакше вивести «No».

Вхідні дані

16

Вихідні дані.

Yes

Задача 2-12

Нехай координати клітин шахової дошки задаються за допомогою чисел, тобто буквені координати «abcd…» змінено на «1234…». Так клітини (1,1) та (1,2) є сусідніми в рядку, а клітини (1,1) та (2,1) – знаходяться по сусідству у стовпчику. Дано координати двох клітинок. Вивести «Yes», якщо вони одного кольору і «No» - якщо клітинки різних кольорів.

Вхідні дані

1 2 1 1

Вихідні дані.

No

Задача 2-13

Дано трьохзначне число. Вивести «Yes», якщо його цифри утворюють зростаючу або спадну послідовність. В іншому випадку вивести «No».

Вхідні дані

122

Вихідні дані.

No

Задача 2-14

Дано координати двох клітинок шахової дошки. Вивести «Yes», якщо слон може за один хід перейти з однієї клітинки в іншу. В іншому випадку вивести «No».

Вхідні дані

1 1 3 3

Вихідні дані.

Yes

Задача 2-15

Дано координати двох клітинок шахової дошки. Вивести «Yes», якщо кінь може за один хід перейти з однієї клітинки в іншу. В іншому випадку вивести «No».

Вхідні дані

3 1 1 2

Вихідні дані.

Yes

Розділ 3. Організація циклів.

Оператори циклу.

Цикл – це форма організації дій, при якій одна і таж послідовність виконується кілька разів доти, поки виконується деяка умова. Розрізняють такі види циклів: цикл з лічильником, цикл з передумовою та цикл з післяумовою.

Цикл з лічильником використовується коли достеменно відомо, скільки разів треба виконувати тіло циклу. Цикл з лічильником складається із заголовка циклу і тіла циклу. Тіло циклу – це оператор, який повторно виконуватиметься. Заголовок – це ключове слово **for**, після якого в круглих дужках записано три вирази, розділені крапкою з комою. Перше вираз обчислюється один раз до початку виконання циклу. Другий – це умова циклу. Тіло циклу повторюватиметься до тих пір, поки умова циклу істина. Третій вираз здійснює приріст лічильника циклу.

Оператора **for** реалізує фундаментальний принцип обчислень в програмуванні – ітерацію. Тіло циклу повторюється для різних, в даному випадку послідовних, значень змінної `i`. Повторення інколи називається ітерацією. Ми нібито проходимо по послідовності значень змінної `i`, виконуючи з поточним значенням одну і ту ж дію, тим самим поступово обчислюючи потрібне значення. З кожною ітерацією ми наближаємося до шуканого значення все ближче і ближче.

Цикли з передумовою та після умовою.

У циклі з передумовою ***спочатку перевіряється умова***, і якщо вона істинна, то тіло циклу виконується у черговий раз, якщо ж умова хибна, то виконання тіла циклу припиняється.

У випадку повторення з після умовою, ***спочатку відбувається виконання тіла циклу***, а після цього перевіряється умова: якщо вона хибна то тіло циклу виконується в черговий раз, а якщо умова істина, то відбувається вихід із циклу.

Можливі ситуації, коли тіло циклу з передумовою не виконується жодного разу. Це відбувається в тому випадку, коли при першій перевірці умови значення логічного виразу є хибним. У цьому полягає відмінність між циклами з передумовою та післяумовою.

Якщо при виконанні циклу умова незмінно залишається істинною, то цикл може виконуватися нескінченно, такий процес називають за циклюванням.

Приклад 3.1

Увести послідовність додатних та від'ємних чисел, що закінчуються нулем. Визначити суму додатних.

```
#include<iostream>
using namespace std;
int main()
      {
            int x,Sum=0;
            do
            {
                  cout<<"x="; cin>>x;
                  if (x>0) Sum=Sum+x;
            }
            while(x!=0);
            cout<<'\n'<<"Sum="<<Sum;
  }
```

Приклад 3.2.

Знайти найбільший спільний дільник натуральних чисел `a` і `b`.

Пояснення: скористаємося алгоритмом Евкліда: будемо зменшувати кожний раз більше число на величину меншого, до тих пір поки ці значення не стануть рівними.

```
#include "stdafx.h"
#include"iostream"
using namespace std;
int main()
{
    unsigned int a,b;
      cout<<"A="; cin>>a;
```

```
      cout<<"B="; cin>>b;
      while (a!=b)
      {
            if(a>b)a=a-b;else b=b-a;
      }
      cout<<"NOD="<<a<<'\n';
    return 0;
}
```

Результат роботи програм не зміниться якщо використати цикл з післяумовою `do…while`:

```
#include "stdafx.h"
#include"iostream"
using namespace std;
int main()
{
    unsigned int a,b;
      cout<<"A="; cin>>a;
      cout<<"B="; cin>>b;
      do
         if(a>b)a=a-b;else b=b-a;
      while (a!=b)
      cout<<"NOD="<<a<<'\n';
    return 0;
}
```

Приклад 1

Підрахувати кількість цифр у десятковому записі невід'ємного числа x.

```
#include<iostream>
using namespace std;
int main()
      {
            int n,x,k=0;
            cout<<"x="; cin>>x;
            n=x;
            do
            {
                  n=n/10;
                  k++;
            }
            while(n!=0);
            cout<<"k="<<k;
    }
```

Приклад 2. Обчислити факторіал натурального числа n.

Пояснення: Факторіалом числа n називаються добуток послідовних чисел, позначається так: n!. За визначенням: $n!=1\cdot 2\cdot ...\cdot (n-1)\cdot n$. Домовлено що факторіал нуля рівний одиниці: 0!=1

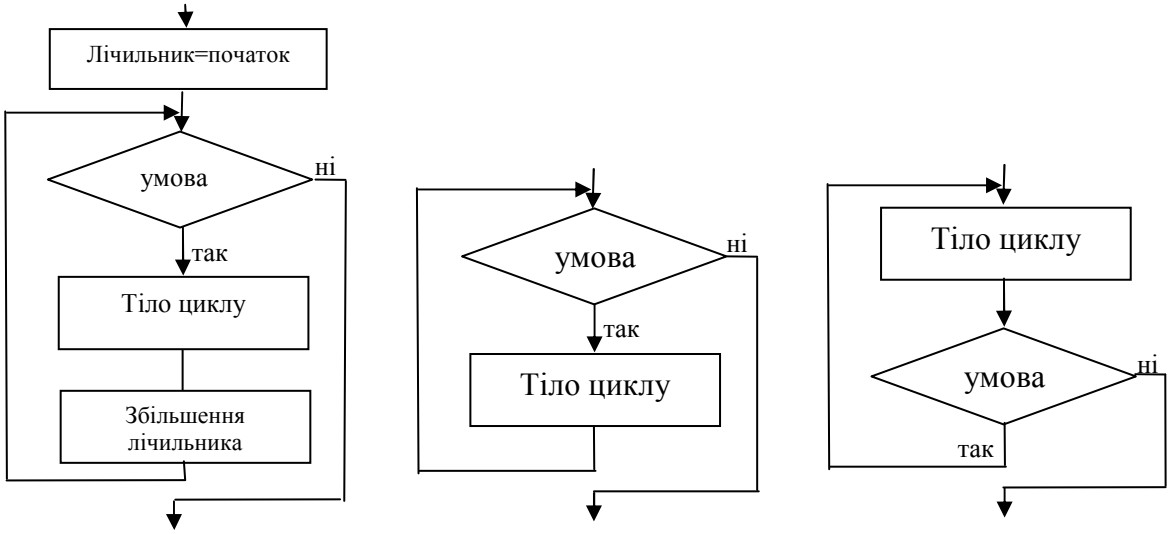
```
#include "stdafx.h"
#include "iostream"
using namespace std;
int main()
{
    unsigned int factorial=1,n;
       cout<<"N="; cin>>n;
       for(int i=2;i<=n;i++){factorial=factorial*i;}
       cout<<"Factorial="<<factorial<<'\n';
}
```

Якщо використати властивості виразів у заголовку оператора **`for`** то дану програму можна записати компактніше:

```
#include "stdafx.h"
#include "iostream"
using namespace std;
int main()
{
    unsigned int factorial=1,n;
```

```
        for(int i=2,cout<<"N=", cin>>n;i<=n;factorial=factorial*i,i++);
        cout<<"Factorial="<<factorial<<'\n';
}
```

Так як перший вираз у заголовку оператора **for** обчислюється один раз, то доцільно у цей вираз через кому записати усі оператори, які мають таку властивість. Третій вираз обчислюється на кожному кроці ітерації, тому доцільно у ньому записати оператори тіла циклу.



Оператори передачі керуваня.

Оператори передачі курування примусово змінюють хід виконання програми. У C++ таких операторів чотири: `goto, break, continue, return.`

Оператор `goto` **`мітка`**, де **`мітка`** звичайний ідентифікатор, застосовують для безумовного переходу, він передає управління операторові з міткою: мітка: оператор.

Оператор `break` здійснює негайний вихід з циклів `while`, `do...while` і `for`, а також з оператора вибору `switch`. Управління передається операторові, що знаходиться безпосередньо за циклом або оператором вибору.

Оператор `continue` починає нову ітерацію циклу, навіть якщо попередня не була завершена.

Оператор `return вираз` завершує виконання функції і передає управління в точку її виклику. Якщо функція повертає значення типу `void`, то вираз в записі оператора відсутній. У противному випадку вираз повинен мати скалярний тип.

Завдання для самоконтролю

Задача 3-1

Дано N цілих чисел. Знайти кількість парних серед них.

ТУ. У першому рядку стандартного вхідного потоку міститься N (N<1000) – кількість чисел. У наступному рядку через пропуск дано самі числа. У стандартний вихідний потік вивести кількість парних чисел.

Вхідні дані

5

12 4 2 7 9

Вихідні дані

3

Задача 3-2

Дано N цілих чисел. Знайти суму трьохцифрових чисел, що є серед даного набору.

ТУ. У першому рядку стандартного вхідного потоку міститься N (N<1000) – кількість чисел. У наступному рядку через пропуск дано самі числа. У стандартний вихідний потік вивести суму трьохцифрових чисел.

Вхідні дані

5

12 4 200 7 101

Вихідні дані

301

Задача 3-3

Дано N цілих чисел. Всі непарні числа піднімати до квадрату та виводити в одному рядку через пропуск.

ТУ. У першому рядку стандартного вхідного потоку міститься N (N<1000) – кількість чисел. У наступному рядку через пропуск дано самі числа. У стандартний вихідний потік вивести квадрати непарних чисел.

Вхідні дані

5

12 4 2 7 9

Вихідні дані

49 81

Задача 3-4

Дано N цілих чисел. Знайти суму чисел кратних 5.

ТУ. У першому рядку стандартного вхідного потоку міститься N (N<1000) – кількість чисел. У наступному рядку через пропуск дано самі числа. У стандартний вихідний потік вивести суму чисел.

Вхідні дані

5

15 4 2 10 9

Вихідні дані

25

Задача 3-5

Дано N цілих чисел. Знайти кількість чисел, що закінчуються двійкою.

ТУ. У першому рядку стандартного вхідного потоку міститься N (N<1000) – кількість чисел. У наступному рядку через пропуск дано самі числа. У стандартний вихідний потік вивести кількість чисел.

Вхідні дані

5

15 4 2 10 92

Вихідні дані

2

Задача 3-6

Знайти всі числа менші N, які є квадратами натуральних чисел.

ТУ. У стандартному вхідному потоці дано ціле додатне N (N<=1000000). У стандартний вихідний потік вивести у порядку зростання відібрані числа.

Вхідні дані

30

Вихідні дані

1 4 9 16 25

Задача 3-7

Знайти дільники числа N.

ТУ. У стандартному вхідному потоці дано ціле додатне N (N<=1000000). У стандартний вихідний потік вивести через пропуск його дільники.

Вхідні дані

10

Вихідні дані.

1 2 5 10

Задача 3-8

Знайти прості дільники числа N.

ТУ. У стандартному вхідному потоці дано ціле додатне N ($N<=10^9$). У стандартний вихідний потік вивести через пропуск його прості дільники. Якщо число N ділиться на деяке просте число більше одного разу, то виводити цей дільник також більше одного разу.

Вхідні дані

20

Вихідні дані.

2 2 5

Задача 3-9

Знайти суму чисел, кратних k і менших n.

ТУ. У стандартному вхідному потоці містяться цілі числа k, n (1<=n,k<=10000). У стандартний вихідний потік вивести суму чисел.

Вхідні дані

7 20

Вихідні дані.

21

Задача 3-10

Знайти середнє арифметичне парних чисел з проміжку [n;m] (1<=n,m<=10000).

ТУ. У стандартному вхідному потоці містяться цілі числа n, m. У стандартний вихідний потік вивести результат з двома знаками після коми.

Вхідні дані

10 13

Вихідні дані.

11.00

Задача 3-11.

Дано натуральне число N (N<=30000). Перевірити чи є воно досконалим. Вивести «Yes» або «No». Число називається досконалим, якщо воно дорівнює сумі всіх своїх додатних дільників, окрім самого себе.

Вхідні дані

6

Вихідні дані.

Yes

Задача 3-12

Дано натуральне число N (N<=30000). Перевірити чи є воно простим. Вивести «Yes» або «No».

Вхідні дані

13

Вихідні дані.

Yes

Задача 3-13

Дано натуральне число N (N<=30000). Вивести всі прості числа не більші за N.

Технічні умови (ТУ). У вхідному потоці дано N, у вихідний потік через пропуск вивести прості числа.

Вхідні дані

13

Вихідні дані.

2 3 5 7 11 13

Задача 3-14

Дано натуральне число N,M (N,M<=30000, N<=M). Вивести всі прості числа з проміжку [N,M].

ТУ: У вхідному потоці дано два числа через пропуск N і M. У вихідний потік через пропуск вивести прості числа.

Вхідні дані

5 10

Вихідні дані.

5 7

Задача 3-15

Дано натуральне число N (N<=30000). Вивести кількість досконалих чисел меньших N.

Вхідні дані
100
Вихідні дані.
2

Задача 3-16
Прості числа, різниця між якими дорівнює два, називають «близнятами». Знайти кількість «близнят», що не більші N.
ТУ: У вхідному потоці дано число N (N<=30000). У вихідний потік вивести кількість «близнят».
Вхідні дані
18
Вихідні дані.
3

Задача 3-17
Дано натуральне число N (N<=30000). Знайти кількість чисел, що є повними квадратами цілих чисел та меншими n.
ТУ: У вхідному потоці дано число N (N<=30000). У вихідний потік вивести кількість чисел - повних квадратів.
Вхідні дані
50
Вихідні дані.
7

Задача 3-18
Дано натуральне число N. Скільки цифр у цьому числі? Задачу розв'язати з використанням операторів організації циклів.
ТУ: У вхідному потоці дано число N (N<=10^9). У вихідний потік вивести кількість цифр.
Вхідні дані
1234
Вихідні дані.
4

Задача 3-19
Знайти кількість чисел виду m^3-3m-3, де m=1, 2, 3….. , не більших n.
ТУ: У вхідному потоці дано число n (n<=3000). У вихідний - вивести кількість таких чисел.
Вхідні дані
10
Вихідні дані.
2

Задача 3-20
Знайти суму чисел виду m^2-2m-1, де m=0, 1, 2,…n.
ТУ: У вхідному потоці дано число n (n<=100). У вихідний - вивести суму чисел.
Вхідні дані
2

Вихідні дані.
-4

Задача 3-21

«Числа Фібоначчі»

Числами Фібоначчі називаються числа послідовності, у якій кожне наступне число дорівнює сумі двох попередніх. Вивести N-те число Фібоначчі (N<=30). Перші два числа послідовності дорівнюють одиниці.

ТУ: У вхідному потоці міститься одне число N, у вихідний потік вивести єдине число, N-те число Фібоначчі.

Вхідні дані

5

Вихідні дані.

5

Задача 3-22

«Кролики»

Є пара кроликів. Пара починає щомісяця народжувати нову пару на третій місяць життя. Скільки кроликів буде через n місяців?

ТУ: У вхідному потоці міститься число n (3<=n<=30), у вихідний потік вивести єдине число – кількість кроликів через n місяців.

Вхідні дані

6

Вихідні дані.

16

Задача 3-23

«Кролики-2»

Земляни знайшли планету, придатну для життя і відправили космічний корабель з одним кроликом щоб переконатися у придатності для життя клімату планети. Кролику сподобався клімат і уже через місяць він привів ще одного (на цій планеті для цього достатньо одного кролика). Дальше кролики почали розмножуватися з такою ж швидкістю – кожен кролик через місяць приводив ще одного. Але, розмноження кроликів почав контролювати монстр із математичними здібностями. Як тільки на початку якогось місяця кроликів ставало більше k, він поїдав k кроликів.

Треба визначити скільки кроликів буде на планеті через n місяців від початку висадки кролика на планету.

ТУ. У вхідному потоці міститься два числа n і k (0<=k,n<=30), у вихідний потік вивести єдине число – кількість кроликів через n місяців.

Вхідні дані

5 10

Вихідні дані

12

Задача 3-24

Задано послідовність із n (0<n<=30000) цілих чисел. Вивести максимальне число, яке зустрічається у цій послідовності.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести найбільше число.

Вхідні дані

7

3 5 1 1 6 6 5

Вихідні дані.

6

Задача 3-25

Задана послідовність із n (0<n<30000) цілих чисел. Вивести порядковий номер останнього максимального числа, яке там зустрічається.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести порядковий номер числа.

Вхідні дані

8

3 5 1 1 0 6 6 5

Вихідні дані.

7

Задача 3-26

Задана послідовність із n (0<n<30000) цілих чисел. Вивести порядковий номер першого максимального числа, яке там зустрічається.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести порядковий номер числа.

Вхідні дані

6

3 5 1 6 6 5

Вихідні дані.

4

Задача 3-27

Задана послідовність із n (0<n<30000) цілих чисел. Вивести кількість елементів, що ідуть підряд і утворюють найбільшу не спадну підпослідовність.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести одне число.

Вхідні дані

7

3 5 1 1 6 6 7

Вихідні дані.

5

Задача 3-28

Задана послідовність із n (0<n<30000) цілих чисел. Вивести кількість елементів, що ідуть підряд і утворюють найбільшу зубчасту підпослідовність. Наприклад, числа 1213243 утворюють зубчасту підпослідовність довжиною 7, за більшим числом іде менше, за меншим – більше.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести одне число.

Вхідні дані
7
3 5 1 1 0 6 5
Вихідні дані.
4

Задача 3-29

Задана послідовність із n (0<n<30000) цілих чисел. Знайти довжину найбільшої підпослідовності, що є монотонною.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести одне число.

Вхідні дані
11
1 1 1 2 3 8 7 5 4 3 1
Вихідні дані.
6

Пояснення. У нашому прикладі є такі монотонні підпослідовності: 111, 238, 875431.

Задача 3-30

Задана послідовність із n (0<n<30000) цілих чисел. Знайти початок та кінець найбільшої підпослідовності, що є монотонною.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести два числа, що є номерами початкового та кінцевого елементів підпослідовності. Якщо таких під послідовностей є декілька, то виводити найменші номери.

Вхідні дані
11
1 1 1 2 3 8 7 5 4 3 1
Вихідні дані.
6 11

Визначення функції – це опис того, як вона працює, тобто які дії треба виконати, щоб отримати шуканий результат. Для функції `sum`, оголошеною вище, визначення може виглядати таким чином:

```
int sum(int a, int b, int c)
{
     int result;
     result = a + b + c;
     return result;
}
```

Перший рядок – це заголовок функції, він збігається з оголошенням функції, за винятком того, що оголошення закінчується крапкою з комою. Далі у фігурних дужках поміщено тіло функції – дії, які дана функція виконує. Аргументи `a, b` і `c` називаються формальними параметрами. Це змінні, які визначені в тілі функції (тобто до них можна звертатися лише усередині фігурних дужок). При написанні визначення функції програма не знає їх значення. При виклику функції замість них підставляються фактичні параметри – значення, з якими функція викликається. Вище, в прикладі виклику функції sum, фактичними параметрами (або фактичними аргументами) були значення змінних `k` , l і `m`. Формальні параметри приймають значення фактичних аргументів, заданих при виклику, і функція виконується. Перше, що ми робимо в тілі функції — оголошуємо внутрішню змінну **`result`** тип якої ціле число. Змінні, оголошені в тілі функції, також називають локальними. Це пов'язано з тим, що змінна **`result`** існує лише під час виконання тіла функції **`sum`**. Після завершення виконання функції вона знищується – її ім'я стає невідомим, і пам'ять, займана цією змінною, звільняється. Другий рядок визначення тіла функції – обчислення результату. Сума всіх аргументів присвоюється змінній **`result`**. Відзначимо, що до присвоєння значення **`result`** було невизначеним (тобто значення змінної було якимсь довільним числом, яке не можна визначити заздалегідь). Останній рядок функції повертає як результат обчислене значення. Оператора `return` завершує виконання функції і повертає вираз, записане після ключового слова `return`, як вихідне значення. У наступному фрагменті програми змінній s присвоюється значення 10:

```
int k = 2;
int l = 3;
int m = 5;
int s = sum(k, l, m);
```

Імена функцій

В мові C++ допустимо мати декілька функцій з одним і тим же ім'ям (перевантаженням імен функцій), тому що функції розрізняються не лише по іменах, але і по типах аргументів. Якщо окрім визначеної вище функції sum ми визначимо ще одну функцію з тим же ім'ям:

```
double sum (double a, double b, double c)
{
     double result;
     result = a + b + c;
     return result;
}
```

це вважатиметься новою функцією. Інколи говорять, що у цих функцій різні підписи. У наступному фрагменті програми вперше буде викликана перша функція, а удруге – друга:

```
int x, y, z, ires;
double p,q,s, dres;
// виклик першого опису функції sum
ires = sum(x,y,z);
// виклик другого опису функції sum
dres = sum(p,q,s);
```

При першому виклику функції `sum` всі фактичні аргументи мають типа `int`. Тому викликається перша функція. У другому виклику всі аргументи мають типа `double`,

відповідно, викликається друга функція. Важливий не лише тип аргументів, але і їх кількість. Можна визначити функцію `sum`, що підсумовує чотири аргументи:

```
int sum(int x1, int x2, int x3, int x4)
{
     return x1 + x2 + x3 + x4;
}
```

Відзначимо, що при визначенні функцій мають значення тип і кількість аргументів, але не тип значення, яке повертається. Спроба визначення двох функцій з одним і тим же ім'ям, одними і тими ж аргументами, але різними значеннями, які повертаються у точку виклику, приведе до помилки компіляції:

```
int foo (int x);
double foo (int x);
// помилка – двічі оголошено одне і теж ім’я
```

Виклик функцій.

Функція являє собою цілком самостійний блок програми. Функція викликається при обчисленні виразів. При виклику їй передаються певні аргументи, функція виконує необхідні дії і повертає результат. Програма на мові С++ містить, принаймні, з одну функцію – функцію **`main()`**. З неї завжди починається виконання програми. Зустрівши ім'я функції у виразі, програма викличе цю функцію, тобто передасть управління на її початок і почне виконувати оператори. Досягнувши кінця функції або оператора return – виходу з функції, управління повернеться в те місце, звідки функція була викликана, підставивши замість неї обчислений результат.

Приклад 1: функція `sqrt` з одним аргументом дійсним числом подвійної точності,повертає результат типа `double`

```
double sqrt(double x);
```

Приклад 2: функція `sum` від трьох цілих аргументів повертає ціле число

```
int sum(int a, int b, int c);
```

Оголошення функції називають інколи прототипом функції. Після того, як функція оголошена, її можна використовувати у виразах:

```
double x = sqrt(3) + 1;
sum(k, l, m) / 15;
```

Якщо функція не повертає жодного результату, тобто вона оголошена як `void`, її виклик не може бути використаний як операнд складнішого виразу, а має бути записаний сам по собі:

```
func(a,b,c);
```

Завдання для самоперевірки

Задача 4-1

Задається N цілих додатних чисел не більших 1000. Знайти суму двоцифрових чисел, що є у даній послідовності.

ТУ. У першому рядку задано число N (N<=1000). У наступному рядку містяться самі числа. У вихідний потік вивести суму чисел, що відповідають умові задачі.

Вхідні дані

5

101 100 10 150 20

Вихідні дані

30

Задача 4-2

Задається N цілих додатних чисел не більших 1000. Знайти суму простих чисел, що є у даній послідовності.

ТУ. У першому рядку задано число N (N<=1000). У наступному рядку містяться самі числа. У вихідний потік вивести суму чисел, що відповідають умові задачі.

Вхідні дані

5

1 2 3 4 5

Вихідні дані

10

Задача 4-3

Задається N цілих додатних чисел не більших 1000. Серед них знайти кількість чисел, що є членами такої послідовності k^2+k+1, де k=0,1,2…..

ТУ. У першому рядку задано число N (N<=1000). У наступному рядку містяться самі числа. У вихідний потік вивести кількість чисел, що відповідають умові задачі.

Вхідні дані

5

3 5 6 7 8

Вихідні дані

2

Задача 4-4

Серед N цілих додатних чисел не більших 1000 знайти два найбільші.

ТУ. У першому рядку задано число N (N<=1000). У наступному рядку містяться самі числа. У вихідний потік вивести два числа через пропуск, що відповідають умові задачі. Спочатку вивести найбільше, а потім наступне за ним по величині.

Вхідні дані

5

1 2 3 4 5

Вихідні дані

5 4

Задача 4-5

Серед N цілих додатних чисел не більших 1000 знайти числа з найбільшою сумою цифр та найменшою. Якщо таких чисел є декілька, то слід вибирати ті, що ідуть у переліку першими.

ТУ. У першому рядку задано число N (N<=1000). У наступному рядку містяться самі числа. У вихідний потік вивести два числа через пропуск, що відповідають умові задачі. Спочатку вивести число з найбільшою сумою цифр, а потім з найменшою.

Вхідні дані

5
12 10 101 1000 102
Вихідні дані
12 10

Задача 4-6
Написати процедуру Minmax(A,B), яка записує у змінну A найменше із значень A та B, а в змінну B — найбільше із цих значень (A і B — дійсні параменти, що є одночасно вхідними та вихідними). Використовуючи чотири виклики процедури знайти найменше та найбільше із чисел A, B, C, D.
Вхідні дані
2 5 3 9
Вихідні дані
2 9

Задача 4-7
Написати процедуру SumDigit(N,S), яка знаходить суму цифр S цілого числа N (N - вхідний, S — вихідний параметр). Використовуючи цю процедуру знайдіть суму цифр для кожного із K даних чисел. Формат вхідних та вихідних даних такий, як у прикладі.
Вхідні дані
5
2 11 20 100 15
Вихідні дані
2
2
2
1
6

Задача 4-8
Дано два прямокутники з сторонами, паралельними осям координат. Знайти точки перетину сторін прямокутників. Гарантується, що лише одна вершина одного прямокутника може лежати всередині іншого і прямокутники не можуть мати більше двох спільних точок.
ТУ. У першому рядку вхідного потоку задаються координати протилежних вершин першого прямокутника, у другому – другого. Координати точок є цілими числами по модулю не більшими 10000. У вихідний потік вивести в першому рядку координату точки перетину, що лежить лівіше , а в другому – координати другої точки, якщо вона є. Якщо ж прямокутники не перетинаються, то вивести -1.
Вхідні дані
0 0 2 2
1 1 3 3
Вихідні дані
1 2
2 1

Задача 4-9
Дано два прямокутники з сторонами, паралельними осям координат. Знайти сумарну площу покриття прямокутниками площини.

ТУ. У першому рядку вхідного потоку задаються координати протилежних вершин першого прямокутника, у другому – другого. Координати точок є цілими числами по модулю не більшими 10000. У вихідний потік вивести площу покриття.

Вхідні дані

0 0 2 2
1 1 3 3

Вихідні дані

7

Задача 4-10

На площині дано N прямокутників і K точок. Які з точок не належать жодному з прямокутників?

ТУ. У першому рядку стандартного вхідного потоку дано N (N<=100). Дальше у N рядках задаються цілі X1,Y1, X2,Y2 (-10000<= X1,Y1, X2,Y2<=10000). Потім задається K (K<=1000) і у наступних K рядках ідуть координати точок X,Y (-10000<=X,Y<=10000). У вихідний потік виводити координати точок, що задовольняють умову задачі у порядку їх переліку.

Вхідні дані

1
0 0 10 10
2
1 1
11 11

Вихідні дані

11 11

Розділ 5. Функції (продовження).

У С++ використовують термін **область видимості змінної**, що визначає частину програми, де ім'я змінної має сенс (отже, може бути використане).
Локальна змінна являє собою змінну, оголошену всередині функції до її використання. Область її видимості: від точки оголошення до кінця даної функції. Досі у своїх програмах ви використовували саме локальні змінні.
При оголошенні локальних змінних всередині функції дуже імовірно, що ім'я локальної змінної, оголошеної вами в одній функції, буде таким же, як й ім'я змінної, використовуваної в іншій функції. Але завдяки тому, що області дії цих змінних обмежені, це не призводить до конфлікту. С++ трактує ім'я кожної змінної як локальне стосовно відповідної функції.
С++ дозволяє у ваших програмах використовувати глобальні змінні, область видимості яких - від точки оголошення до кінця програми (глобально для всіх функцій). Оголошувати глобальну змінну слід поза усіма функціями (зокрема, поза функцією **main):**

```
float global_var;                    //глобальна змінна
int main() { // оператори програми }
```

Глобальні змінні застосовують для передачі даних між функціями, але це утруднює пошук помилок у програмі. Для обміну даними між функціями використовують параметри функцій і значення, як функція повертає.

Передача параметрів у функцію.
Обмін інформацією між функціями здійснюється за допомогою механізму передачі параметрів. Список змінних, вказаний в заголовку функції називається формальними параметрами або просто параметрами функції. Список змінних в операторові виклику функції - це фактичні параметри, або аргументи. Механізм передачі параметрів забезпечує заміну формальних параметрів фактичними параметрами і дозволяє виконувати функцію з різними даними. Між фактичними параметрами в операторові виклику функції і формальними параметрами в заголовку функції встановлюється взаємно однозначна відповідність, тобто кількість, тип та порядок формальних і фактичних параметрів повинні співпадати.
Передача параметрів виконується таким чином. Обчислюються вирази, що стоять на місці фактичних параметрів. У пам'яті виділяється місце під формальні параметри, відповідно до їх типів. Потім формальним параметрам присвоюють значення фактичних. Виконується перевірка типів і при необхідності виконується їх перетворення. При невідповідності типів видається діагностичне повідомлення.
Передача параметрів у функцію може здійснюватися за значенням і за адресою. При передачі даних за значенням функція працює з копіями фактичних параметрів, і доступу до вихідних значень аргументів у неї немає. При передачі даних за адресою у функцію передається не змінна, а її адреса, і, отже, функція має доступ до елементів пам'яті, в яких зберігаються значення аргументів. Таким чином, дані, передані за значенням, функція змінити не може, на відміну від даних, переданих по адресу.
Якщо потрібно заборонити зміну параметра усередині функції, використовують модифікатор **const**. Заголовок функції в загальному вигляді виглядатиме так:

```
тип імя_функції(const тип змінної* імя_змінної,…)
```

Випадкові числа
Заголовковий файл **<stdlib.h>** містить функцію **random**(x), яка повертає випадкове ціле число в діапазоні від 0 до x. Перед першим звертанням до функції **random** рекомендується викликати функцію **randomize(),** яка ініціалізує генератор випадкових чисел. Якщо цього не зробити, то при кожному запуску програми будемо мати однакову послідовність випадкових чисел:

```
#include<iostream.h>                    //Програма    4.7
#include<conio.h> # include< s tdlib.h> int main()
```

hi
randomize ();
,for(int i=0;i<10;i++)
cout<<random(999)<<" "; getch();return 0; } ,

Випадкові числа широко використовуються при написанні тестуючих та ігрових програм.

Повернення результату за допомогою оператора return

Повернення результату з функції у точку виклику здійснюється оператором

```
return(вираз);
```

Працює оператор таким чином. Обчислюється значення виразу, вказаного після **return** і перетвориться до типу значення, яке повертається. Виконання функції завершується, а обчислене значення передається у точку виклику. Будь-які оператори, які записані у функції за оператором **return**, ігноруються. Програма продовжує свою роботу з оператора наступного за оператором виклику даної функції. Також функція може містити декілька операторів **return**, якщо це визначено потребами алгоритму. Наприклад, в наступній програмі функція f порівнює значення змінної з нулем. Якщо число позитивне, то у програму передається значення 1, якщо від'ємне, то -1, а якщо число є нулем, то 0.

Завдання для самоконтролю.

Задача 5-1

Знайти найбільший спільний дільник даних чисел.

ТУ. У першому рядку задано число N (0<N<1000), у наступному рядку через пропуск задаються самі цілі числа не більші 2*10^9. У вихідний потік вивести число, що є найбільшим спільним дільником для даних чисел.

Вхідні дані

4

10 22 30 50

Вихідні дані

2

Примітка. Можна використати таке співвідношення: НСД(a,b,c)=НСД(НСД(a,b),c).

Задача 5-2

Знайти значення виразу:

$$\frac{4\cdot\sum_{k=n}^{2n}2k}{n+\sum_{k=n-1}^{3n}2k}+\sum_{k=2n}^{3n}2k$$

ТУ. У стандартному вхідному потоці міститься число N (0<N<=1000). У вихідний потік вивести значення виразу з двома знаками після коми.

Вхідні дані

1

Вихідні дані

11.85

Задача 5-3

Описати функцію Calc(A, B, Op) дійсного типу, яка буде виконувати одну операцію над ненульовими дійсними A і B. Операція визначається цілим параметром Op: 1 - віднімання, 2 - множення, 3 - ділення, будь-які інші значення – додавання. З допомогою цієї функції для кожної із N (N<1000) трійок чисел A, B, Op вивести результат операції з точністю до двох знаків після коми.

Вхідні дані

3

2 3 1

3 2 2

2 4 3

Вихідні дані

-1.00

6.00

0.50

Задача 5-4

Описати функцію IsSquare(K) логічного типу, яка повертає True, якщо цілий параметр K (K > 0) є повним квадратом цілого числа, і False в іншому випадку. З допомогою цієї функції для N (N<1000) даних натуральних чисел визначити кількість чисел, що є повними квадратами цілих чисел.

Вхідні дані

3

4 15 36

Вихідні дані

2

Задача 5-5
Описати функцію DigitCount(K) цілого типу, яка знаходить кількість цифр цілого додатного числа K (K<10^18). З допомогою цієї функції для кожного з N (N<1000) чисел K вивести його кількість цифр.
Вхідні дані
3
4 1521 36009
Вихідні дані
1
4
5

Задача 5-6
Знайти на проміжку [N,M] кількість простих чисел, які можна розбити ще на два простих числа. До таких чисел належать, наприклад числа: 23 (2 і 3), 137 (13 і 7), 739 (7 і 39).
ТУ. У стандартному вхідному потоці знаходяться числа N,M (20<N,M<50000). У вихідний потік вивести кількість чисел, що задовольняють умову задачі.
Вхідні дані
21 40
Вихідні дані
2

Задача 5-7
Серед даних чисел знайти кількість чисел Фібоначчі. Числами Фібоначчі називаються числа, перші два з яких дорівнюють одиниці, а кожне наступне рівне сумі двох попередніх.
Наприклад: 1, 1, 2, 3, 5, 8,…
ТУ. У першому рядку задано число N (0<N<10000). У наступному рядку ідуть самі числа не більші 2*10^9. У вихідний потік вивести кількість чисел Фібоначчі.
Вхідні дані
5
1 18 3 4 5
Вихідні дані
3

Задача 5-8
Серед даних чисел знайти числа з найбільшою та найменшою сумою цифр. Якщо таких чисел є декілька то виводити ті, що ідуть першими у переліку.
ТУ. У першому рядку задано число N (0<N<10000). У наступному рядку ідуть самі числа не більші 2*10^9. У вихідний потік вивести спочатку число з найменшою сумою цифр, а потім через пропуск – з найбільшою.
Вхідні дані
5
1 18 31 41 54
Вихідні дані
1 18

Задача 5-9
Дано N натуральних чисел. У кожному числі переставити цифри таким чином, щоб утворене число було мінімально можливе. Цифра «0» не може бути першою цифрою числа.

ТУ. У першому рядку стандартного вхідного потоку знаходиться число N (N<=100000). Дальше у N рядках по одному числу, що не перевищують 2*10^9. У вихідний потік вивести числа по одному у кожному рядку, що задовольняють умову задачі.
Вхідні дані
5
1230
395
10
987
9078
Вихідні дані
1023
359
10
789
7089

Задача 5-10
Для заданих N натуральних чисел знайти їх найменше спільне кратне. Гарантується, що найменше спільне кратне не буде перевищувати 10^18.
ТУ. У першому рядку стандартного вхідного потоку знаходиться число N (N<=100). Дальше у N рядках по одному числу, що не перевищують 2*10^9. У вихідний потік вивести ціле число – найменше спільне кратне..
Вхідні дані
3
7
11
13
Вихідні дані
1001

Розділ 6. Масиви

Масиви.

Масив – сукупність елементів одного типу. Прикладом елементів одного типу може бути: цілі числа, рядки, дати, символи і т.д. Наприклад, можна створити масив для зберігання списку учнів вашої групи. Замість створення змінних для кожного учня, наприклад *Учень1*, *Учень2* і так далі, досить створити один масив, де кожному прізвищу із списку буде присвоєно порядковий номер. Таким чином:

Масив - структурований тип даних, що складається з фіксованого числа елементів одного типа.

Табл. 1 Одномірний числовий масив

12.7	0.13	-1.5	0	21.9	-3.7	5.0	121.7
0-й елемент масиву	1-й елемент масиву	2-й елемент масиву	3-й елемент масиву	4-й елемент масиву	5-й елемент масиву	5-й елемент масиву	6-й елемент масиву

Масив має 7 елементів, кожен з яких зберігає число дійсного типу. Елементи в масиві пронумеровані. Такого роду масив, що є просто списком даних одного і того ж типу, називають простим або одновимірним масивом. Для доступу до даних, що зберігаються в певному елементі масиву, необхідно вказати ім'я масиву і порядковий номер цього елементу, який називають індексом.

Опис масиву.

Описати масив можна так:

```
тип ім'я_масиву [розмірність];
```

де розмірність - це кількість елементів в масиві. Наприклад:

```
int  x[10];        //описує масив  x  із 10  цілих чисел
float  a [10];    //описує масив  із  20  дійсних чисел.
```

Розмірність масиву та тип його елементів визначають об'єм пам'яті, який необхідний для зберігання масиву, тому розмірність - це цілий додатній константний вираз. Наприклад:

```
const  int n=15;    //описана ціла додатна константа
double B[n];    //описаний масив з  15  дійсних чисел
```

Елементи масиву в С++ нумеруються з нуля. Перший елемент завжди має номер нуль, а номер останнього елементу на одиницю менше заданої при його опису розмірності:

```
Char C[5] // описаний масив символів, елементи якого нумеруються від 0 до 5.
```

Основні операції над масивами.

Доступ до кожного елементу масиву здійснюється за допомогою індексу - порядкового номера елементу. Для звернення до елементу масиву вказують його ім'я, а потім в квадратних дужках індекс, наприклад:

```
const  int n=15;
double B[n], Sum;
Sum=B[0]+B[n-1] ; //сума першого та останнього елементів масиву.
```

Масиву, як і будь-якій іншій змінній, можна присвоїти початкове значення (***ініціалізувати***). Для цього значення елементів масиву потрібно перерахувати у фігурних дужках через кому:

```
float   a[6]= {1.2,(float)3/4,5./6,6.1};    //формується масив із шести дійсних чисел, значення елементам присвоюється по порядку,елементи,значення яких не вказані, обнуляються.
```

Значення елементів масиву будуть такі: a[0]=1.2, a[l]=0.75, a[2]=0.8333, a[3]=6.1, a[4]=0, a[5]=0, для елементів a[1] та a[2] виконується перетворення типів – зверніть на це увагу.

Всі маніпуляції з масивами в С++ здійснюються поелементно. Організовується цикл, в якому відбувається послідовне звернення до нульового, першого, другого і так далі елементу масиву.

Ввід та вивід елементів масиву.

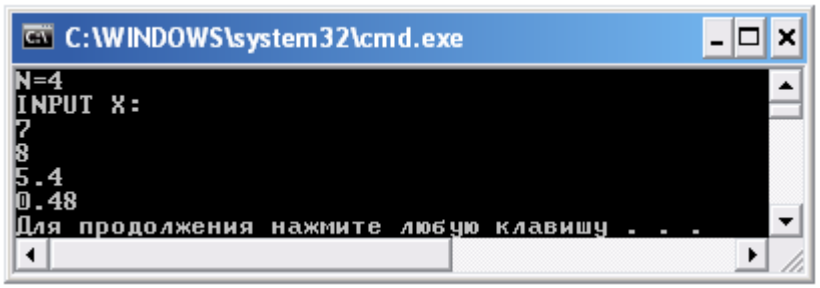
Ввід та вивід елементів масиву також можна здійснювати поелементно. Розглянемо декілька варіантів воду елементів масиву:

Приклад 6.1: ввід масиву за допомогою функції **`scanf()`**:

```
#include "stdafx.h"
#include<stdio.h>
using namespace System;
```

```
int main()
{
    float x[10]; int i,n;
      printf("N="); scanf ("%d",&n);// ввід розмірності масиву
      printf("INPUT X: \n");
      for(i=0;i<n;i++)
            scanf("%f",&x[i]);
    return 0;
}
```

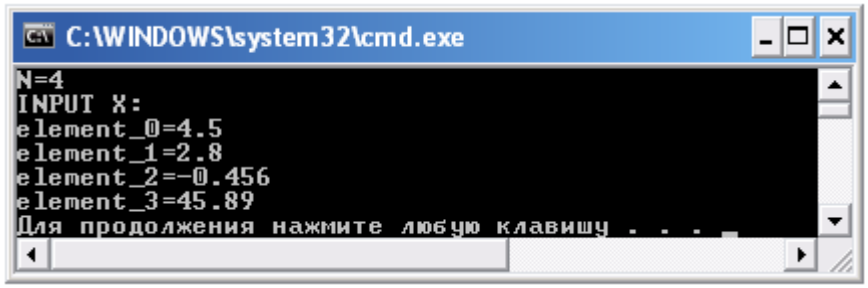
Результат виконання програми:



Приклад 6.2: ввід масиву за допомогою функції **`scanf()`** та допоміжної змінної ***b***:

```
#include "stdafx.h"
#include<stdio.h>
using namespace System;
int main()
{
    float x[10],b; int n;
      printf("N="); scanf("%d",&n);// ввід розмірності масиву
      printf("INPUT X: \n");
      for(int i=0;i<n;i++)
      {
            printf("element_%d=",i);
            scanf("%f",&b);
            x[i]=b;
      }
    return 0;
}
```

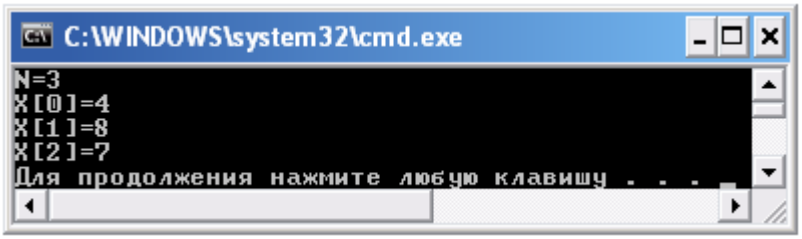
Результат виконання програми:



Приклад 6.3: ввід масиву за допомогою оператора **`cin`**:

```
#include "stdafx.h"
#include<iostream>
using namespace std;
int main()
{
    int x[10],n;
      cout<<"N="; cin>>n;// ввід розмірності масиву
      for(int i=0;i<n;i++)
      {
            cout<<"X["<<i<<"]="; //вивід повідомлення про ввід
            cin>>x[i];//ввід елемента масиву
      }
```

```
        return 0;
    }
```
Результат виконання програми:



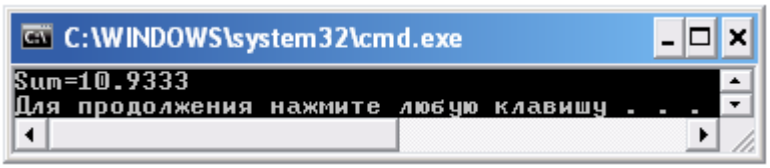
Вивід елементів масиву можна здійснювати декількома способами:

Варіант 1:
```
for(int i=0;i<n;i++) printf("%f \t",x[i]);// вивід у вигляді рядка
```
Варіант 2:
```
for(int i=0;i<n;i++) printf("\n %f",x[i]);// вивід у вигляді стовпця
```
Варіант 3:
```
for(int i=0;i<n;i++) cout<<"x["<<i<<"]="<<x[i]<<'\t';// вивід у вигляді рядка
```
Варіант 4:
```
for(int i=0;i<n;i++) cout<<"x["<<i<<"]="<<x[i]<<'\n';// вивід у вигляді стовпця
```

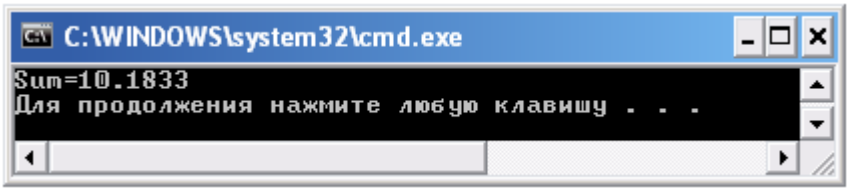
Обчислення суми елементів масиву.

Заданий масив x, який містить шість дійсних чисел. Знайти суму елементів масиву.

Приклад 6.4
```
#include "stdafx.h"
#include<iostream>
using namespace std;
int main()
{
        const int n=6; double sum=0;
        double x[n]={1.2,3./4,5./6,6.1,1.8,0.25};
        for(int i=0;i<n;sum+=x[i],i++);// обчислення суми
        cout<<"Sum="<<sum<<'\n';
}
```
Результат виконання програми:



Творче завдання: у попередній програмі забрано один символ і результат виконання став таким:



Який символ видалили і чому змінився результат?

Завдання 6.1.

Що потрібно змінити у попередній програмі, щоб вона обчислювала добуток шести дійсних чисел?

Пошук максимального елемента в масиві.

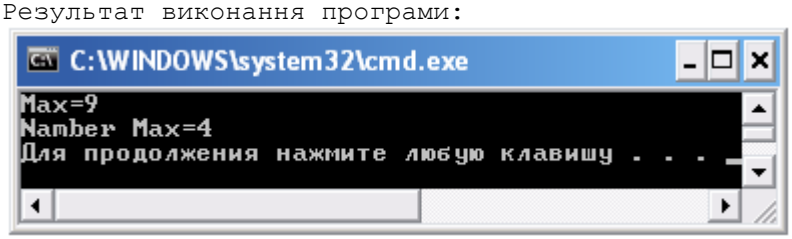
Заданий масив **mas** який містить n елементів. Знайти максимальний елемент масиву та його номер.

Алгоритм розв'язку задачі наступний: нехай у змінній з іменем **max** зберігається значення максимального елемента, а у змінній **Nmax** – його номер. Припустимо, що максимальним є

нульовий елемент. Далі, у циклі починаючи з першого, порівнюємо елементи масиву із максимальним. Якщо, на деякому кроці, ми знайдемо елемент, який більший за максимальний, то запишемо його у змінну **max**, а його номер – у змінну **Nmax**. Даний алгоритм реалізований у програмі:

```
#include "stdafx.h"
#include<iostream>
using namespace std;
int main()
{
    const int n=7; int max,Nmax=0,i;
      int x[n]={4,7,3,8,9,2,5};
      for(max=x[0],i=1;i<n;i++)
            if (max<x[i])
            {
                  Nmax=i;
                  max=x[i];
            }
      cout<<"Max="<<max<<'\n';
      cout<<"Namber Max="<<Nmax<<'\n';
}
```

Результат виконання програми:



Творче завдання: результат програми не зміниться, якщо провести такі зміни:

```
#include "stdafx.h"
#include<iostream>
using namespace std;
int main()
{
    const int n=7; int Nmax=0,i;
      int x[n]={4,7,3,8,9,2,5};
      for(i=1;i<n;i++) if (x[Nmax]<x[i])Nmax=i;
      cout<<"Max="<<x[Nmax]<<'\n';
      cout<<"Namber Max="<<Nmax<<'\n';
}
```

Поясніть логіку роботи програми.

Видалення елемента масиву.

Нехай нам потрібно видалити із масиву **x** елемент під номером ***k***. Для цього нам достатньо записати ***k+1*** на місце ***k-го***, ***k+2*** на місце ***k+1*** і так далі і наостанок останній елемент з номером ***n-1*** записати на місце елемента з номером ***n-2***. У подальшому при роботі з цим масивом потрібно врахувати, що кількість елементів зменшилась на одиницю.

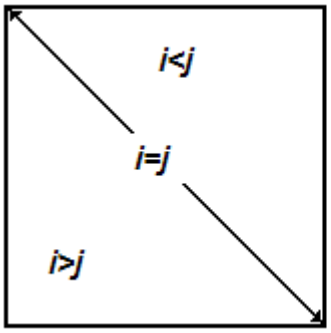
```
#include "stdafx.h"
#include<iostream>
using namespace std;
int main()
{
    const int n=7; int k=4,i;//
      int x[n]={4,7,3,8,9,2,5};
      for(i=k;i<n-1;x[i]=x[i+1],i++);//видалення к-го елемента
      for(i=0;i<n-1;i++)cout<<x[i]<<'\t';cout<<'\n';//вивід елементів масиву
}
```

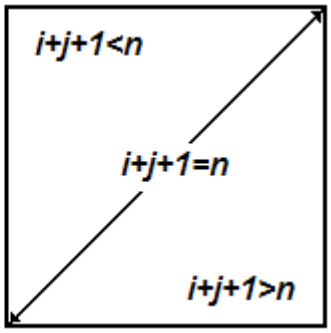
Багатомірні масиви.

Масиви в С++ можуть бути також двовимірними, тривимірними й більше. Простим прикладом двовимірного масиву є сторінка класного журналу, що містить інформацію в рядках і стовпцях. Такий масив має два індекси: перший указує номер рядка, а другий - номер стовпця, а тому для роботи із двомірними масивами використовують два вкладених цикли

Двомірні масиви називають ***матрицями***. Матриці мають такі властивості:

- Якщо номер рядка елемента співпадає з номером стовпця ***(i=j),*** це означає що елемент лежить на головній діагоналі;
- Якщо номер рядка перевищує номер стовпця ***(i>j),*** то елемент знаходиться нижче головної діагоналі;
- Якщо номер рядка перевищує номер стовпця ***(i<j),*** то елемент знаходиться вище головної діагоналі;
- Елемент належить бічній діагоналі, якщо його індекси задовольняють умову: ***i+j+1=n;***
- Нерівність ***i+j+1<n*** характерна для елемента, який знаходиться вище бічної діагоналі;
- Нерівність ***i+j+1>n*** характерна для елемента, який знаходиться нижче вище бічної діагоналі;





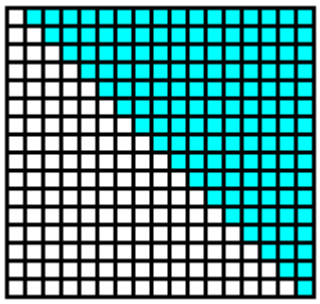
Масив з ***n*** рядками й ***m*** стовпцями називають масивами розміром ***n×m*** (наприклад, масив 3 на 4). Кожен елемент у масиві **a** визначається ім'ям елемента, у формі ***a*[i][j]; *a*** - це ім'я масиву, а ***i*** та ***j*** - індекси, які однозначно визначають кожен елемент в ***a***. Нумерація рядків і стовпців елементів масиву починається з нуля. Багатовимірні масиви можуть одержувати початкові значення у своїх оголошеннях так само, як масиви з єдиним індексом. Наприклад, двовимірний масив **c[3] [2]** можна оголосити й дати йому початкові значення в такий спосіб:

```
int c[3][2]={{4, 2},{6, 7},{5,8}};
```

визначення групуються в рядки, вміщені у фігурні дужки. Таким чином, елементи `c[0][0]` і `c[0][1]` одержують початкові значення 4 й 2, а елементи `c[1][0]` і `c[1][1]` одержують початкові значення 6 й 7 і т.д. Якщо початкових значень у даному рядку не вистачає для їхнього присвоєння всім елементам рядка, то елементам, що залишаються, присвоюються нульові початкові значення. Таким чином, оголошення

```
int c[2][2]={{10},{9,14}};
```

буде означати що `c[0][0]` одержує початкове значення 10,. `c[0][1]` одержує початкове значення 0, `c[1][0]` одержує початкове значення 9 і `c [1] [1]` одержує початкове значення 14.



Приклад 6.2 Знайти суму елементів матриці розміром ***n×m,*** які лежать вище головної діагоналі (див рис.).

Алгоритм розв'язку даної задачі побудований таким чином: у чарунку `s`, де буде зберігатися сума, на початку заносимо 0, а потім за допомогою двох циклів проглядаємо кожний елемент матриці, сумуємо елементи при умові i<j.

Сортування елементів масиву.

Завдання для самоконтролю.

Задача 6-1

Дана лінійна таблиця розмірності N. Знайти суму від'ємних елементів.

ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – суму від'ємних чисел.

Вхідні дані

5

2 -3 -6 1 2

Вихідні дані

-9

Задача 6-2

Дана лінійна таблиця розмірності N. Знайти кількість непарних чисел.

ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – кількість непарних чисел.

Вхідні дані

5

2 -3 -6 1 2

Вихідні дані

2

Задача 6-3

Дана лінійна таблиця розмірності N, нумерація елементів починається з 1. Знайти кількість непарних чисел, що мають парні індекси.

ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – кількість чисел.

Вхідні дані

5

2 3 5 1 2

Вихідні дані

2

Задача 6-4

Дана лінійна таблиця розмірності N, нумерація елементів починається з 1. Знайти індекс останнього максимального числа.

ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – найбільший індекс максимального числа.

Вхідні дані

5

2 3 7 1 7

Вихідні дані

5

Задача 6-5

Дана лінійна таблиця розмірності N, нумерація елементів починається з 1. Знайти індекс першого входження максимального числа.

ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – мінімальний індекс максимального числа.

Вхідні дані

5

2 3 7 1 7
Вихідні дані
3

Задача 6-6
Дана лінійна таблиця розмірності N, нумерація елементів починається з 1. Знайти суму індексів всіх непарних чисел.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – сума індексів.
Вхідні дані
5
2 3 7 1 7
Вихідні дані
14

Задача 6-7
Дана лінійна таблиця розмірності N. Вивести спочатку всі парні числа, а потім – непарні.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести через пропуск N чисел відповідно до умови задачі.
Вхідні дані
5
1 2 3 4 5
Вихідні дані
2 4 1 3 5

Задача 6-8
Дана лінійна таблиця розмірності N. Вивести ці числа у оберненому порядку.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести через пропуск N чисел відповідно до умови задачі.
Вхідні дані
5
1 2 3 4 5
Вихідні дані
5 4 3 2 1

Задача 6-9
Дана лінійна таблиця розмірності N. Вивести «Yes», якщо числа впорядковані за зростанням і «No» - в іншому випадку.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести «Yes» або «No».
Вхідні дані
5
1 2 3 4 5
Вихідні дані
Yes

Задача 6-10
Дана лінійна таблиця A розмірності N. Вивести кількість таких індексів i, для яких виконується така умова: A[i]>A[j] для всіх j<i.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести кількість індексів.

Вхідні дані
5
1 1 1 4 5
Вихідні дані
3
Примітка. Перший елемент рахуємо як такий, що відповідає умові задачі.

Задача 6-11
Дана лінійна таблиця розмірності N. Впорядкувати ці числа за зростанням.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести через пропуск N впорядкованих чисел.
Вхідні дані
5
3 5 3 4 1
Вихідні дані
1 3 3 4 5

Задача 6-12
Дана лінійна таблиця розмірності N. Знайти найбільшу кількість елементів, що повторюються.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести через пропуск два числа: число і кількість його повторів. Якщо таких чисел є кілька варіантів, то виводити те, що зустрілося у таблиці перше.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані
3 4

Задача 6-13
Дана лінійна таблиця розмірності N. Назвемо «близькими» такі числа, що модуль різниці їх індексів та модуль різниці самих чисел дорівнюють одиниці. Знайти кількість пар таких чисел.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести кількість пар.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані
3

Примітка. Пари «близьких» чисел в порядку їх слідування в таблиці: (3,4), (1,2), (2,3).

Задача 6-14
Дана лінійна таблиця розмірності N. Знайти кількість пар «близьких» чисел та самі пари цих чисел.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік у першому рядку вивести K - кількість пар. У наступних K рядках виводимо по два числа через пропуск – пари «близьких» чисел.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані

3
3 4
1 2
2 3

Задача 6-15
Дана лінійна таблиця розмірності N. Знайти та вивести всі прості числа, що є в даній таблиці.
ТУ. У першому рядку вхідного потоку задано N<10000, у другому через пропуск N цілих чисел не більших 100000. У вихідний потік у першому рядку вивести кількість К простих чисел, у другому – через пропуск самі прості числа.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані
6
3 5 3 2 3 3

Задача 6-16
Дана лінійна таблиця розмірності N. Знайти та вивести всі прості числа, що є в даній таблиці. Ті числа, що повторюються враховувати лише один раз.
ТУ. У першому рядку вхідного потоку задано N<10000, у другому через пропуск N цілих чисел не більших 30000. У вихідний потік у першому рядку вивести кількість різних простих чисел, у другому – через пропуск ці числа.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані
3
3 5 2

Задача 6-17
Дана лінійна таблиця розмірності N. Знайти та вивести кількість елементів більших за середнє арифметичне цих чисел..
ТУ. У першому рядку вхідного потоку задано N<10000, у другому через пропуск N цілих чисел. У вихідний потік у першому рядку вивести кількість К таких чисел, у другому – через пропуск самі числа.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані
6
3 5 3 4 3 3

Задача 6-18
Дана матриця розмірності $N \times M$. Вивести рядок з максимальною сумою елементів.
ТУ. У першому рядку вхідного потоку міститься два числа N,M (0<N,M<=100). Дальше у N рядка через пропуск міститься по М чисел. У вихідний потік вивести рядок з М чисел з максимальною сумою.
Вхідні дані
3 4
4 3 1 1
1 1 1 2

5 6 3 1
Вихідні дані
5 6 3 1

Задача 6-19
Дана матриця розмірності NxN. Знайти суму елементів, що знаходяться вище головної діагоналі..
ТУ. У першому рядку вхідного потоку міститься число N (0<N<=100). Дальше у N рядка через пропуск міститься по N чисел. У вихідний потік вивести суму.
Вхідні дані
3
4 3 1
1 1 1
5 6 3
Вихідні дані
5

Задача 6-20
Дана матриця розмірності $N \times M$. Знайти суму «внутрішніх» елементів матриці. «Внутрішніми» елементами ми назвемо ті, що не знаходяться на крайньому рядку чи стовпцю.
ТУ. У першому рядку вхідного потоку міститься два числа N,M (0<N,M<=100). Дальше у N рядках через пропуск міститься по M чисел. У вихідний потік вивести суму або 0 якщо таких елементів немає.
Вхідні дані
3 4
4 3 1 1
1 1 1 2
5 6 3 1
Вихідні дані
2

Задача 6-21
Дана матриця розмірності NxN. Перевірити, чи є вона магічним квадратом. Магічним квадратом називається матриця, у якої сума рядків, стовпців та діагоналей рівна.
ТУ. У першому рядку вхідного потоку міститься N (0<N<=100). Дальше у N рядка через пропуск міститься по N чисел. У вихідний потік вивести „Yes” або „No”.
Вхідні дані
3
1 1 1
1 1 1
1 1 1
Вихідні дані
Yes

Задача 6-22
Дана лінійна таблиця розмірності N (N<=100). Яку найменшу кількість елементів треба видалити, щоб утворилася зростаюча послідовність.
ТУ. У вхідному потоці дано N+1 ціле число: перше число N, а за ним через пропуск слідують інші числа. У вихідний потік вивести кількість чисел, які треба видалити.
Вхідні дані
5 1 1 2 2 3

Вихідні дані
2

Задача 6-23
Дано координати точок на площині. Вивести кількість різних трикутників, які можна отримати взявши дані точки за вершини.
ТУ. У першому рядку міститься число N (3<=N<100). У наступних N рядках – пари чисел, що є координатами точок Xi,Yi (-1000<=Xi,Yi<=1000). У вихідний потік вивести кількість трикутників.
Вхідні дані
4
-1 1
0 0
1 1
0 2
Вихідні дані
4

Задача 6-24
Дана матриця розмірності $N \times N$, елементами якої є 0 та 1. Групу одиничок, що межує по горизонталі та по вертикалі з нулями назвемо «островом». Знайти кількість «островів».
ТУ. У першому рядку вхідного потоку міститься N (0<N<=100). Дальше у N рядка через пропуск міститься по N чисел. У вихідний потік вивести кількість «островів».
Вхідні дані
4
1 0 0 1
0 1 1 0
1 1 1 1
1 1 1 1
Вихідні дані
3

Задача 6-25
Дана матриця розмірності NxN, елементами якої є 0 та 1. Знайти площу найбільшого «острова».
ТУ. У першому рядку вхідного потоку міститься N (0<N<=100). Дальше у N рядка через пропуск міститься по N чисел. У вихідний потік вивести ціле число – площу «острова».
Вхідні дані
4
1 0 0 1
0 1 1 0
1 1 1 1
0 0 0 0
Вихідні дані
6

Задача 6-26
Лабіринт задано матрицею з 0 та 1 розмірності NxM. Нулі показують проходи, а одинички перепони. Знайти довжину найкоротшого шляху з комірки (1,1) в комірку (N,M). Рухатися можна лише по горизонталі або вертикалі.

ТУ. У першому рядку вхідного потоку містяться N, M (0<N,M<=100). Дальше у N рядка через пропуск міститься по M чисел. У вихідний потік вивести довжину шляху або -1, якщо його не існує.

Вхідні дані

4 5
0 0 0 1 0
0 1 0 1 0
0 1 0 0 0
0 1 1 1 0

Вихідні дані

7

Задача 6-27

Дано N відрізків прямої. Знайти довжину їх спільної частини.

ТУ. У першому рядку вхідного потоку міститься N (0<N<=100). У наступних рядках вводиться по два цілих числа: координати лівого та правого кінця відрізку. Ліва координата завжди не більша правої. У вихідний потік вивести довжину спільної частини, або -1 якщо такої немає. Якщо спільною є лише точка, то виводити 0.

Вхідні дані

3
1 10
3 15
2 6

Вихідні дані

3

Задача 6-28

Дано N відрізків прямої. Знайти сумарну довжину частин прямої, покритої хоча би одним відрізком.

ТУ. У першому рядку вхідного потоку міститься N (0<N<=100). У наступних рядках вводиться по два додатні числа не більші 1000: координати лівого та правого кінця відрізку. Ліва координата завжди менша правої. У вихідний потік вивести довжину покриття.

Вхідні дані

3
1 10
11 12
2 6

Вихідні дані

10

Задача 6-29

Дано N цілих чисел. Треба знайти таких три числа, щоб їх добуток був максимальним.

ТУ. У першому рядку дано число N (3<=N<=100). У наступному рядку дано N чисел, кожне з яких по модулю не перевищує 1000. У вихідний потік вивести три числа у порядку їх слідування в таблиці.

Вхідні дані

9
10 3 5 1 7 9 0 9 -3

Вихідні дані

10 9 9

Задача 6-30

У початковий момент в лінійний масив записані числа від 1 до N (на і-му місці знаходиться число і). З елементами масиву проробляють таку операцію: міняють місцями числа, що мають індекси і та j.

ТУ. У першому рядку дано число N (3<=N<=100), у другому рядку ціле число K (0<K<10000) – кількість операцій, дальше у K рядках знаходяться по два числа – індекси елементів масиву. Вивести в рядок всі елементи масиву після K таких операцій.

Вхідні дані

10
2
1 3
5 3

Вихідні дані

3 2 5 4 1 6 7 8 9 10

Розділ 7. Масиви символів

Рядком називается послідовність символів, які закінчуються нуль-байтом. Для работи з рядками (типом string) слід підключити бібліотеку <**string**>:

```
#include <string>
using namespace std;
```

Створення рядка відбувається одним із наступних прийомів:

```
string a;           // Створення нуль-рядка
string b("Hello");  // Створення рядка з початковою ініціалізацією
string c(10,'a');   // Створення рядка, який містить 10 букв 'a'
string d = b;       // Створюється рядок d і у нього копіюється вміст рядка b
string e(b,1,2);    // У створений рядок e копіюється 2 символа рядка b,
                    //  начинаючи з позиції 1. Значення створеного рядка e рівно "el"
```

Конкатенація. Опрерація '+' використовується для об'єднання (конкатенація) рядків:

```
string a("This");
string b("cat");
string c = a+b;  // Значення c рівно "Thiscat"
```

Ітератори. Ітератор *string*::*begin*() вказує на початок, а *string*::*end*() на кінець рядка.

```
string a("ABCDEFGH");
string b(a.begin()+1,a.end()-1); // b = "BCDEFG"
```

Ввід-вивід. При використанні функцій бібліотеки <**cstdio**> для вводу-вивода рядків слід безпосередньо перед операцією переводити їх в массив символів. Це здійснюєтья функцією ***c_str*()**. Вивід рядка символів:

```
string a("Hello");
printf("%s\n",a.c_str());
```

Ввід рядка при допомозі функції ***scanf*** слід здійснювати у символьний масив. Далі масив символів можна перетворювати в тип *string* при допомозі конструктора:

```
char s[100];
string a;
scanf("%s",s); a = string(s);
printf("%s\n",a.c_str());
```

Форматований ввід із рядка здійснюється функцією ***sscanf***.
Приклад: із рядка вибрати числа та знайти їх суму:

```
int x,y;
string a("23+4");
sscanf(a.c_str(),"%d+%d",&x,&y);
printf("%d+%d=%d\n",x,y,x+y);
```

Нумерація індексів елементів рядка починається з нуля. *i* - ий символ рядка *a* можна отримати операцією індексації *a*[*i*] або виконаням функції *at*(*i*):

```
string a("This is a hat");
printf("Second letter is %c or %c\n",a[1],a.at(1));
```

Розмір рядка *a* рівний ***a.size*().**

Приклад: здійснити посимвольний вивід рядка *a*:

```
string a("This is a hat");
for(i=0;i<a.size();i++) printf("%c",a[i]);
```

Підрядки. Якщо *a* – зміна типу **string**, то ***a.substr(pos, len)*** є підрядком, який починається з позиції ***pos*** та має довжину ***len***. Якщо другий аргумент відсутній, то виділяється підрядок починаючи з позиції ***pos*** і до кінця рядка. Наприклад:

```
string a("This is a hat");
string b = a.substr(2,4);                    // b = "is i"
string c = a.substr(3);                      // c = "s is a hat"
printf("%s\n%s\n",b.c_str(),c.c_str());
```

Функция ***is_prefix*** повертає 1, якщо рядок *a* є префіксом рядка *b*.

```
int is_prefix(string a, string b)
{
  return b.substr(0,a.size()) == a;
}
```

Пошук. Якщо ***a*** і ***b*** – змінні типу string, то ***a.find(b)*** повертає першу позицію входження рядка ***b*** у рядок ***a***. Якщо *b* не є підрядком *a*, то метод ***find*** повертає -1.

```
string a = "qwertwqy";
string b = "qw";
int pos = a.find(b);                  // pos = 0
```

Використовуючи метод *find*, функцію is_prefix можна переписати так:

```
int is_prefix(string a, string b)
{
  return (b.find(a) == 0);
}
```

Пошук в рядку s підрядка b, що починається не раніше ***i-ої*** позиції, слід здійснювати за допомогою методу find(b, i) класу string:

```
string s = "abcdabcabc";
int pos = s.find("ab",1);             // pos = 4
```

Видалення підрядків. Якщо ***t*** є підрядком ***s***, то знаходимо позицію ***i***, з якою вона починається і видаляємо за допомогою методу `erase`.

```
string s = "Hello thisis world";
string t = "this";
int i = s.find(t);
s.erase(i,t.size());                  // s = "Hello is world"
```

Видалення всіх входжень рядків ***t*** як підрядки в рядку ***s*** можна зробити в циклі:

```
string s = "thisHello thisis world this isthis";
string t = "this";
while((i = s.find(t)) >= 0)
  s.erase(i,t.size());                // s = "Hello is world  is"
```

Розбиття. Разбиение строки *s* на несколько подстрок при помощи разделителя *c* совершается при помощи функции split. Разделитель *c* может встречаться в строке в любом месте и в любом количестве.

Розбиття рядка ***s*** на декілька підрядків за допомогою роздільника ***c*** здійснюється за допомогою функції **`split`**. Роздільник ***c*** може зустрічатися в рядку в будь-якому місці і в будь-якій кількості.

```
vector<string> split(string s,char c)
{
  vector<string> res;
  int i=0,j = s.find(c);
  while(j >= 0)
```

```
  {
    if (s[i] != c) res.push_back(s.substr(i,j-i));
    i = ++j;
    j = s.find(c,j);
  }
  res.push_back(s.substr(i));
  return res;
}
```

Виклик функції і виведення результату:

```
string s = "  g  this     is a test    9";
vector<string> v = split(s,' ');
for(i=0;i<v.size();i++) printf("%s ",v[i].c_str());
```

Заміна. Приклад заміни всіх входжень пропусків в рядку ***v*** на крапку:

```
string s = "This is a text";
replace(s.begin(),s.end(),' ','.');  // s = "This.is.a.text"
```

Реверс рядка. Реверс рядка можна зробити за допомогою функції **reverse**, оголошеною в <**algorithm**>:

```
s = "This";
reverse(s.begin(),s.end());          // s = "sihT"
```

Потокове введення із рядка. Занесення чисел, що містяться в рядку s, в масив m, можна зробити за допомогою потокового виводу:

```
#include <cstdio>
#include <string>
#include <sstream>
using namespace std;
int main()
{
  string s = "2   34  55       9 111";
  int m[10],ptr=0,i,a;
  stringstream in(s);
  while (in >> a) m[ptr++] = a;
  for(i=0;i<ptr;i++) printf("%d ",m[i]); printf("\n");
}
```

Перетворення рядків і чисел. Для перетворення рядка в число і назад скористаємося шаблоном cvt.

```
#include <cstdio>
#include <sstream>
#include <string>
using namespace std;
template<class A, class B>
  A cvt(B x)
  {
    stringstream s;s<<x;
    A res;
    s>>res;
    return res;
  }
int main(void)
{
  int a    = cvt<int>("123");
  string b = cvt<string>(123);
  printf("%d %s\n",a,b.c_str());
  return 0;
}
```

Завдання для самоконтролю.
Якщо в тексті наступних задачі не вказано довжину символьного рядка, то його довжину вважати не більшою 255 символів.

Задача 7-1
Дано рядок символів. Знайти кількість слів у даному рядку. Слова розділяються одним пропуском.
Ввід <
I love you!
Вивід >
3

Задача 7-2
Дано рядок символів. Знайти кількість слів у даному рядку. Слова розділяються довільною кількістю пропусків.
Ввід <
I love you!
Вивід >
3

Задача 7-3
Дано рядок символів. Визначити кількість літер латинського алфавіту.
Ввід <
I love you!
Вивід >
8
Задача 7-4
Дано рядок символів. Побудувати обернений рядок.
Ввід <
abcd
Вивід >
dcba

Задача 7-5
Дано рядок символів. Перевірити чи є він паліндромом. Паліндромом називаються рядки, що однаково читаються зліва направо і справа наліво. Вивести «Yes» або «No».
Ввід <
abba
Вивід >
Yes

Задача 7-6
Дано рядок символів. Знищити в ньому всі пропуски.
Ввід <
a b b a
Вивід >
abba

Задача 7-7
Дано рядок символів. Після кожного символу рядка вставити пропуск та вивести.
ТУ. У стандартному вхідному потоці міститься рядок довжиною не більше 127 символів.
Ввід <

abba
Вивід >
a b b a

Задача 7-8
Дано рядок символів. Скопіювати цілу частину половини символів, що знаходяться на початку рядка та «приклеїти» їх в кінець.
ТУ. У стандартному вхідному потоці міститься рядок довжиною не більше 127 символів.
Ввід <
abcde
Вивід >
abcdeab

Задача 7-9
Дано рядок символів. Додати до кінця даного рядка рядок, обернений до нього.
ТУ. У стандартному вхідному потоці міститься рядок довжиною не більше 127 символів.
Ввід <
abc
Вивід >
abccba

Задача 7-10
Дано рядок символів. Добавити в початок даного рядка рядок, обернений до нього.
ТУ. У стандартному вхідному потоці міститься рядок довжиною не більше 127 символів.
Ввід <
abc
Вивід >
cbaabc

Задача 7-11
Дано рядок символів. Над символами рядка виконуємо наступну операцію: переглядаємо символи починаючи з першого і кожен другий копіюємо в кінець. Виводимо результат.
ТУ. У стандартному вхідному потоці міститься рядок довжиною не більше 127 символів.
Ввід <
abc
Вивід >
abcbb

Задача 7-12
Дано рядок символів. Над символами рядка виконуємо наступну операцію: переглядаємо символи починаючи з першого і кожен другий переносимо в кінець. Виводимо результат.
Ввід <
abcd
Вивід >
acbd

Задача 7-13
Дано рядок символів. Перевірити чи є в даному рядку символи «13». Вивести «Yes» або «No».
Ввід <
abcd123abc

Вивід >
No

Задача 7-14
Дано рядок символів. Впорядкувати символи даного рядка за зростанням.
Ввід <
fedba
Вивід >
abdef

Задача 7-15
Дано рядок символів. Визначити кількість малих літер латинського алфавіту.
Ввід <
Abba
Вивід >
3

Задача 7-16
Дано рядок символів. Визначити кількість цифр у рядку.
Ввід <
Ab’ba1o o4
Вивід >
2

Задача 7-17
Дано рядок символів, що містить один символ «-». У першому рядку вихідного потоку вивести частину рядка до символу «-», у другому рядку – частину, що знаходиться після цього символу. Символ «-» не виводити в жодній частині.
Ввід <
Abba-Yes
Вивід >
Abba
Yes

Задача 7-18
Дано рядок виду: a#b=, де a та b деякі цілі додатні числа не більші 10000, а символ «#» - одна із операцій: «+», «-», «*». Знайти значення виразу s та у вихідний потік вивести рядок a#b=s.
Ввід <
2+3=
Вивід >
2+3=5

Задача 7-19
Дано рядок символів, що містить числа. У вихідний потік вивести всі числа по одному в кожному рядку.
Ввід <
Abba1980 Yes5NO1990 Ok2 5!
Вивід >
1980
5
1990
2

5

Задача 7-20
Дано рядок символів, що містить числа. У вихідний потік вивести суму цих чисел.
ТУ. Числа та сума не перевищують $2*10^9$.
Ввід <
Abba1980 Yes5NO1990 Ok2 5!
Вивід >
3982

Задача 7-21
Рядок містить не менше двох слів. Впорядкувати слова в алфавітному порядку.
ТУ. У вхідному потоці міститься рядок слів розділених пропусками. У вихідний потік вивести впорядковані слова по одному слову в рядок.
Ввід <
f e d b a
Вивід >
a
b
d
e
f

Задача 7-22
Дано два символьних рядки. Чи можна за допомогою символів першого рядка скласти другий рядок. Кожен символ першого рядка можна використовувати не більше частоти його входжень у другий рядок. У вихідний потік вивести «Yes» або «No».
Ввід <
abcdefedcba
abba
Вивід >
Yes

Задача 7-23
Дано два символьних рядки, що містять лише символи латинського алфавіту. Чи можна другий рядок отримати шляхом перестановки символів першого рядка. У вихідний потік вивести «Yes» або «No».
Ввід <
abcd
dbac
Вивід >
Yes

Задача 7-24
Дано два символьних рядки. Чи є перший рядок підрядком другого? Вивести позицію першого входження або 0 у випадку, коли такого підрядка не існує. Якщо входжень є декілька, то вивести позицію першого входження.
Ввід <
abcd
abcabcddd
Вивід >
4

Задача 7-25
Дано два символьних рядки. Знайти кількість входжень першого підрядка в другий. У стандартний вихідний потік вивести одне число – кількість входжень.
Ввід <
abca
abcabca
Вивід >
2

Задача 7-26
Дано послідовність слів, що розділяються пропусками. Треба вивести перші літери всіх слів. Якщо перша буква мала, то виводити її великою. При виконанні цього завдання слова “and”, “the”, “of” будемо ігнорувати.
ТУ. У стандартному вхідному потоці містяться слова, що складаються із літер латинського алфавіту. У стандартний вихідний потік вивести в одному рядку великі літери, що відповідають умові задачі.
Ввід <
the united states of america
Вивід >
USA

Задача 7-27
Дано речення, що закінчується крапкою в кінці. Вивести частоту входження символів латинського алфавіту у даний текст. Виводити лише символи, що зустрілися в тексті хоча би один раз. Формат виведення має бути зрозумілим із прикладу. Великі та малі літери не розрізняти.
Ввід <
Abcdaabcd.
Вивід >
a-3
b-2
c-2
d-2

Задача 7-28
Дано речення, що закінчується крапкою в кінці. Вивести символ, що повторюється у тексті найчастіше. У випадку наявності кількох таких символів, вивести той, що перший іде в алфавіті. Великі та малі літери не розрізняти.
ТУ. У стандартному вхідному потоці міститься речення, що закінчується крапкою. У стандартний вихідний потік вивести у першому рядку символ, у другому – кількість його входжень.
Ввід <
Abcdaabcd.
Вивід >
a
3

Задача 7-29
Нам відома система кодування слів, що називається «голосні латинські». Всі голосні букви видаляються із слова і дописуються в його кінець в тому ж порядку, що є вони у слові. Голосні букви є: ‘a’, ‘e’, ‘i’, ‘o’, ‘u’. Треба закодувати слово за такою системою.

ТУ. У стандартному вхідному потоці міститься слово довжиною не більше 255 символів, що складається лише з великих та малих літер латинського алфавіту. У вихідний потік вивести закодоване слово.

Вхід <
Application
Вивід >
pplctnAiaio

Задача 7-30
В заданому тексті знайти середню довжину слова. Слова складаються з букв латинського алфавіту та можуть розділятися цифрами, пропусками та символами пунктуації.
ТУ. У стандартному вхідному потоці міститься текст довжиною не більше 30000 символів, що закінчується символом «#». У стандартний вихідний потік вивести дійсне число з точністю до десятих.

Вхід <
This is div easy problem.#
Вивід >
4.0

Задача 7-31
Дано символьні рядки, що можуть містити числа. Вивести ці числа через пропуск. Крапка в тексті може бути лише у десяткових числах.
ТУ. У вхідному потоці містяться рядки довжиною не більше 255 символів. Кількість рядків не більша 1000. У вихідний потік вивести числа через пропуск у відповідних рядках.

Вхідні дані
Julja 25, Vasja 30.3
2+3=5

Вихідні дані
25 30.3
2 3 5

Задача 7-32
Дано текст довжиною не більше 30000 символів. Всі слова, що починаються з великої латинської літери вивести в окремому рядку великими літерами. Слово може складатися і з однієї літери, а самі слова розділяються будь-яким символом, що не є літерою латинського алфавіту.
ТУ. У вхідному стандартному потоці міститься текст. У вихідний потік вивести по одному у рядку великими літерами слова, що задовольняють умову задачі.
Вхідні дані
Hi-Tech

Вихідні дані
HI
TECH

Задача 7-33

Задано N слів, що складаються з малих латинських літер. Знайти кількість слів, що починаються та закінчуються голосною літерою.
ТУ. У першому рядку стандартного вхідного потоку знаходиться N (0<N<1001), у наступних N рядках міститься по одному слову. У вихідний потік вивести кількість слів, що відповідають умові задачі.
Вхідні дані
4
abbi
then
else
o
Вихідні дані
3

Задача 7-34
Задано N слів, що складаються з малих латинських літер. Знайти кількість слів, що містять хоча би дві однакові літери.
ТУ. У першому рядку стандартного вхідного потоку знаходиться N (0<N<1001), у наступних N рядках міститься по одному слову. У вихідний потік вивести кількість слів, що відповідають умові задачі.
Вхідні дані
4
abbi
then
else
o
Вихідні дані
2

Задача 7-35
Задано N слів, що складаються з малих латинських літер. Знайти кількість слів, що містять лише різні літери.
ТУ. У першому рядку стандартного вхідного потоку знаходиться N (0<N<1001), у наступних N рядках міститься по одному слову. У вихідний потік вивести кількість слів, що відповідають умові задачі.
Вхідні дані
4
abbi
then
else
o
Вихідні дані
2

Задача 7-36
Задано N слів, що складаються з малих латинських літер. Знайти кількість слів, що є паліндромами. Паліндромами називається слова, що однаково читаються зліва направо і справа наліво.
ТУ. У першому рядку стандартного вхідного потоку знаходиться N (0<N<1001), у наступних N рядках міститься по одному слову. У вихідний потік вивести кількість слів, що відповідають умові задачі.

Вхідні дані
4
abba
then
else
o
Вихідні дані
2

Задача 7-37
Задано N слів, що складаються з малих латинських літер. Знайти кількість слів, літери яких не порушують зростаючий алфавітний порядок.
ТУ. У першому рядку стандартного вхідного потоку знаходиться N (0<N<1001), у наступних N рядках міститься по одному слову. У вихідний потік вивести кількість слів, що відповідають умові задачі.
Вхідні дані
4
abb
then
else
aabbccddzz
Вихідні дані
2

Розділ 8. Рекурсивні функції та процедури

Рекурсією називається такий спосіб організації обробки даних, при якому програма (або функція) викликає сама себе або безпосередньо, або з інших програм (функцій).

Ітерацією називається такий спосіб організації обробки даних, при якому деякі дії багаторазово повторюються, і не використовують рекурсивний виклик програм (функцій).

Теорема. *Довільний алгоритм, реалізований в рекурсивній формі, може бути поданий в ітераційній формі і навпаки.*

Розглянемо набір елементарних функцій, які реалізовані як при допомозі операторів циклу, так і при допомозі рекурсивного підходу. Перед написанням рекурсивних програм (функцій) на будь-якій мові програмування, як правило, необхідно записати ***рекурентний вираз***, який визначає метод обчислення функції. Рекурентний вираз повинен містити як мінімум дві умови:

I) умова продовження рекурсії (крок рекурсії);

II) умова виходу із рекурсії.

Рекурсію будемо реалізовувати безпосереднім викликом функції самої себе. При цьому в тілі функції слід перевірити умову продовження рекурсії. Якщо умова істина, то виходимо із функції, інакше здійснюємо рекурсивний крок.

Ітеративний варіант функції будемо реалізовувати при допомозі оператора циклу **for**.

Основні задачі на рекурсію.

1. Факторіал числа. Факторіалом цілого невід'ємного числа n називається добуток всіх натуральних чисел від 1 до n і позначається $n!$. Якщо $f(n) = n!$, то має місце рекурентне співвідношення:

$$\begin{cases} f(n) = n * f(n-1), \\ f(0) = 1 \end{cases}$$

Перша рівність описує крок рекурсії – метод обчислення $f(n)$ через $f(n-1)$. Друга рівність вказує умову виходу із рекурсії. Якщо її не вказати, то функція буде працювати нескінченно довго (за циклювання).

Наприклад, значення $f(3)$ можна обчислити наступним чином:

$$f(3) = 3 * f(2) = 3 * 2 * f(1) = 3 * 2 * 1 * f(0) = 3 * 2 * 1 * 1 = 6$$

Зрозуміло, що при обчисленні $f(n)$ слід здійснити n рекурсивних викликів.

рекурсивна реалізація	***циклічна реалізація***
C++	

`int f(int n)`
`{`
`  if(!n) return 1;`
`  return n * f(n - 1);`
`}`
	`int f(int n)`
	`{`
	`  int i, res = 1;`
	`  for(i = 1; i <= n; i++) res = res * i;`
	`  return res;`
	`}`

Ідея циклічної реалізації полягає у безпосередньому обчисленні факторіала числа при допомозі оператора циклу:

$$f(n) = 1 * 2 * 3 * \ldots * n$$

2. Степінь числа за лінійний час. Обчислення степеня числа $f(a, n) = a^n$ за лінійний (O(n)) час можна здійснити при допомозі наступних рекурентного співвідношення:

$$\begin{cases} f(a,n) = a * f(a,n-1), \\ f(a,0) = 1 \end{cases}$$

рекурсивна реалізація	***циклічна реалізація***

`int f(int a,int n)`
`{`
`  if (!n) return 1;`
`  return a * f(a, n - 1);`
`}`
	`int f(int a,int n)`
	`{`
	`  int i, res = 1;`
	`  for(i = 0; i < n; i++) res = res * a;`
	`  return res;`
	`}`

В ітераційному варіанті достатньо обчислити добуток: $a * a * \ldots * a$.

3. Степінь числа за логарифмічний час. Обчислення степеня числа $f(a, n) = a^n$ за $O(\log_2 n)$ здійснюється на основі наступного рекурентного співвідношення:

$$\begin{cases} f(a,n) = a * f(a^2, \lfloor n/2 \rfloor), n \text{ непарне} \\ f(a,n) = f(a^2, \lfloor n/2 \rfloor), n \text{ парне} \\ f(a,0) = 1 \end{cases}$$

Наприклад, підносячи у десяту степінь можно реалізувати так:

$$a^{10} = \left(a^5\right)^2 = \left(a \cdot \left(a^2\right)^2\right)^2$$

Оскільки піднесення до квадрату еквівалентно одному множенню, то для обчислення a^{10} достатньо здійснити 4 операції множення.

рекурсивна реалізація	***циклічна реалізація***
C++	

`int f(int a, int n)`
`{`
`  if (!n) return 1;`
`  if (n & 1) return a * f(a * a, n / 2);`
`  return f(a * a, n / 2);`
`}`
	`int f(int a, int n)`
	`{`
	`  int  res = 1;`
	`  while(n > 0)`
	`  {`
	`  if (n & 1) res *= a;`
	`  n >>= 1; a *= a;`
	`  }`
	`  return res;`
	`}`

4. Сума цифр числа. Суму цифр натурального числа n можна знайти при допомозі функції $f(n)$, яка визначена наступним чином:

$$\begin{cases} f(n) = n \bmod 10 + f(n/10), \\ f(0) = 0 \end{cases}$$

Умова продовження рекурсії: сума цифр числа рівна останній цифрі плюс сума цифр числа без останньої цифри (числа, поділеного націло на 10).

Умова закінчення рекурсії: якщо число рівно 0, то сума його цифр рівна 0.

Наприклад, сума цифр числа 234 буде обчислюватися наступним чином:

$$f(234) = 4 + f(23) = 4 + 3 + f(2) = 4 + 3 + 2 + f(0) = 4 + 3 + 2 + 0 = 9$$

рекурсивна реалізація	***циклічна реалізація***
C++	

`int f(int n)`
`{`
`  if (!n) return 0;`
`  return n % 10 + f(n / 10);`
`}`
	`int f(int n)`
	`{`
	`  int res = 0;`
	`  for(; n>0; n = n / 10) res = res + n % 10;`
	`  return res;`
	`}`

5. Число одиниць. Кількість одиниць у двійковому представлені числа n можна знайти при допомозі функції $f(n)$, яка визначається наступним чином (& - операція побітового ‘і’):

$$\begin{cases} f(n) = 1 + f(n \,\&\, (n-1)), \\ f(0) = 0 \end{cases}$$

В результаті операції $n = n \,\&\, (n-1)$ пропадає остання одиниця у двійковому представлені числа n:

$$n = a_1 a_2 \ldots a_{k-1} a_k 10 \ldots 0$$
$$n-1 = a_1 a_2 \ldots a_{k-1} a_k 01 \ldots 1$$
$$n \,\&\, (n-1) = a_1 a_2 \ldots a_{k-1}\, 000 \ldots 0$$

Рекурсивний виклик функції f буде здійснюватися стільки раз, скільки одиниць у двійковому представлені числа n.

рекурсивна реалізація	***циклічна реалізація***

`int f(int n)`
`{`
`  if (!n) return 0;`
`  return 1 + f(n & (n - 1));`
`}`
	`int f(int n)`
	`{`
	`  int res = 0;`
	`  for(; n > 0; n = n & (n - 1)) res++;`
	`  return res;`
	`}`

6. Біноміальний коефіцієнт. Значення біноміального коефіцієнта обчислюється за формулою:

$$C_n^k = \frac{n!}{k!(n-k)!}$$

і визначається рекурентним співвідношенням:

$$C_n^k = \begin{cases} C_{n-1}^{k-1} + C_{n-1}^k, n > 0 \\ 1, k = n \text{ или } k = 0 \end{cases}$$

```
int c(int k, int n)
{
  if (n == k) return 1;
  if (k == 0) return 1;
  return c(k - 1, n - 1) + c(k, n - 1);
}
```

Враховуючи, що $C_n^k = \frac{n(n-1)\ldots(n-k+1)}{1 \cdot 2 \cdot \ldots \cdot k}$, значення біноміального коефіцієнта можна обчислити при допомозі циклу. При цьому всі операції ділення будуть цілочисельними. Якщо $k > n-k$, то слід використати співвідношення $C_n^k = C_n^{n-k}$ щоб запобігти Time Limit при обчисленні, наприклад, значення $C_{1000000001}^{1000000000}$.

```
int Cnk(int k, int n)
{
  long long res = 1;
  if (k > n - k) k = n - k;
  for(int i = 1; i <= k; i++)
    res = res * (n - i + 1) / i;
  return (int)res;
}
```

7. Рекурсивна функція. Для заданого натурального n обчислити значення функції $f(n)$, яка задана рекурентними співвідношеннями:

$$f(2 * n) = f(n),$$
$$f(2 * n + 1) = f(n) + f(n+1),$$
$$f(0) = 0, f(1) = 1$$

Програмна реалізація функції $f(n)$ має вигляд:

```
int f(int n)
{
  if (n <= 1) return n;
  if (n % 2) return f(n / 2) + f(n / 2 + 1);
  return f(n / 2);
}
```

При такій реалізації деякі значеня функції f можуть обчислюватися декілька раз. Розглянемо інший підхід до обчислення значеня f. Визначимо функцію

$$g(n, i, j) = i * f(n) + j * f(n + 1),$$

для якої має місце рівність:

$$g(2 * n, i, j) = g(n, i + j, j),$$
$$g(2 * n + 1, i, j) = g(n, i, i + j),$$
$$g(0, i, j) = i * f(0) + j * f(1) = j$$

Використовуючи наведені вирази, можно обчислити значеня $f(n) = g(n, 1, 0)$ із часовою оцінкою O(log n).

```
int g(int n, int i, int j)
{
  if (!n) return j;
  if (n % 2) return g(n / 2, i, i + j);
  return g(n / 2, i + j, j);
}

int f(int n)
{
  return g(n, 1, 0);
}
```

8. Функція Акермана. Функція Акермана A(m, n) визначається рекурсивним виразом:

$$A(0, n) = n + 1,$$
$$A(m, 0) = A(m - 1, 1),$$
$$A(m, n) = A(m - 1, A(m, n - 1))$$

Рекурсивна реалізація функції Акермана має вигляд:

```
int a(int m, int n)
{
  if (!m) return n + 1;
  if (!n) return a(m - 1, 1);
  return a(m - 1, a(m, n - 1));
}
```

Для малих значень m функцію Акермана можні виразити явно:

$$A(0, n) = n + 1$$
$$A(1, n) = 2 + (n + 3) - 3 = n + 2$$
$$A(2, n) = 2 * (n + 3) - 3 = 2 * n + 3$$
$$A(3, n) = 2^{n+3} - 3$$

9. НСД.(найбільший спільний дільник)

Знайти НСД використавши наступну рекурентність:

$$\text{НСД}(a, b) = \begin{cases} a, b = 0 \\ \text{НСД}(b, a \bmod b), \text{b} \neq 0 \end{cases}$$

Якщо $a < b$, то НСД(a, b) = НСД(b, a mod b) = НСД(b, a), тобто аргументи функції НСД переставляються. При наступних викликах функції НСД перший аргумент завжди більший за другий. Нулем може стати лише другий аргумент b.

Реалізуємо мовою програмування Сі функцію gcd (*Greatest Common Divisor*) обчислення НСД, використовуючи останню наведену рекурентність. Знак % в Сі позначає операцію взяття остачі від ділення.

```
int gcd(int a, int b)
{
  if (b == 0) return a;
  return gcd(b, a % b);
}
```

Нагадаємо, що ***умовний оператор*** в мові Сі має наступний синтаксис:

```
if (<умовний вираз>) <вираз 1>; else <вираз 2>;
```

Якщо <умовний вираз> є істинним, то виконується <вираз 1>, інакше виконується <вираз 2>.

Тернарний умовний оператор має наступний синтаксис:

< умовний вираз > ? < вираз 1 > : < вираз 2 >;

та семантично дещо відрізняється від оператора ***if..then..else***. Якщо < умовний вираз > є істинним, то оператор повертає значення, яке повертає < вираз 1 >, інакше повертається значення виразу < вираз 2 >.

Використовуючи тернарний оператор, функцію gcd можна записати наступним чином:

```
int gcd(int a, int b)
{
  return (!b) ? a : gcd(b, a % b);
}
```

10. НСК(найменше спільне кратне)

Теорема. Між НСД та НСК двох чисел має місце наступне співвідношення:

$$a * b = \text{НСД}(a, b) * \text{НСК}(a, b)$$

Функція lcm (*Lowest Common Multiple*) обчислення НСК має вигляд:

```
int lcm(int a, int b)
{
  return a / gcd(a, b) * b;
}
```

Помітимо, що при обчисленні виразу $a * b / \gcd(a, b)$ можливе переповнення, а при обчисленні $a / \gcd(a, b) * b$ ні. Вважаємо, що значення a, b та $\text{lcm}(a, b)$ знаходяться в межах типу int.

Завдання для самоконтролю

Задача 8-1

Дано функцію піднесення до степеня.

$$a^n = \begin{cases} 1, & n = 0 \\ a^{n-1} * a, & n > 0 \end{cases}$$

Знайти значення функції, для даних a та n.

ТУ. У стандартному вхідному потоці дано цілі a та n (0<a<10, 0<=n<50). У вихідний потік вивести значення функції. Гарантується, що результат не буде перевищувати 10^{18}.

Вхідні дані

2 4

Вихідні дані

16

Задача 8-2

Знайти значення функції

$$Y(n) = \begin{cases} 2, & n=1 \\ 2*Y(n-1) \end{cases}$$

ТУ. У стандартному вхідному потоці дано ціле n (0<n<50). У вихідний потік вивести значення функції.

Вхідні дані

2

Вихідні дані

4

Задача 8-3

Знайти найбільший спільний дільник двох чисел A, B. Написати рекурсивну функцію або процедуру.

ТУ. У вхідному потоці задається через пропуск два цілих числа не більших 10^9. У вихідний потік вивести їх найбільший спільний дільник.

Вхідні дані

6 9

Вихідні дані

3

Задача 8-4

Послідовність чисел задається таким рекурентним співвідношенням:

G(n)=2*G(n-2), G(0)=0, де G(1)=1.

Для даного n знайти G(n).

ТУ. У вхідному потоці дано ціле n (0<=n<100). У вихідний потік вивести n-й член послідовності.

Вхідні дані

3

Вихідні дані

2

Задача 8-5

Обчислити значення функції:

$$G(n) = \begin{cases} 0, & n=0 \\ & \\ & \end{cases}$$

G(n-1)+2, n>0

ТУ. У вхідному потоці задається ціле n ($0<=n<10^4$). У вихідний потік вивести значення функції.
Вхідні дані
3
Вихідні дані
6

Задача 8-6
Знайти всі дільники числа N, які є числами Фібоначчі.
ТУ. У вхідному потоці дано N ($N<10^6$). У вихідний потік через пропуск вивести дільники у порядку зростання.
Вхідні дані
10
Вихідні дані
1 2 5

Задача 8-7
Знайти значення «91-функції Мак-Карті», як задається таким чином:

$$F(n)=\begin{cases} n-10, & n>100 \\ F(F(n+11)), & n<=100 \end{cases}$$

ТУ. У вхідному потоці задається ціле додатне N<200. У вихідний потік вивести значення функції.
Вхідні дані
199
Вихідні дані
189

Задача 8-8
Знайти значення функції, що задається таким чином:

$$S(n) = \begin{cases} n, & n<10 \\ S(n \text{ div } 10) + n \bmod 10, & n>=10 \end{cases}$$

ТУ. У вхідному потоці задається ціле додатне N ($N<=10^9$). У вихідний потік вивести значення функції.
Вхідні дані
11
Вихідні дані
2

Задача 8-9
Для заданого натурального числа, вивести його цифри у зворотному порядку.
ТУ. У вхідному потоці задається натуральне N, що містить не більше 200 цифр. У вихідний потік вивести число з оберненим порядком цифр.
Вхідні дані
1230

Вихідні дані
0321

Задача 8-10
Знайти значення функції для n , якщо

$$T(n) = \begin{cases} 1, & n=1 \\ 2*T(n \text{ div } 2), & n>1 \end{cases}$$

ТУ. В одному рядку вхідного потоку задаються n (n<=100000). Вхідні дані підібрані таким чином, що результат можна отримати. У вихідний потік вивести значення коефіцієнта.
Вхідні дані
10
Вихідні дані
8

Задача 8-11
Знайти біноміальний коефіцієнт для даних m,n (0<=n<=m), якщо

$$C(m,n) = \begin{cases} 1, & n=m,\ n=0,\ m<=1 \\ C(m-1,n-1) + C(m-1,n) \end{cases}$$

ТУ. В одному рядку вхідного потоку задаються m, n (m,n<30). У вихідний потік вивести значення коефіцієнта.
Вхідні дані
10 2
Вихідні дані
45

Задача 8-12
Розбиттям натурального числа називається його представлення у вигляді суми натуральних чисел. Суми з різним порядком доданків вважаються однаковими. Знайти кількість розбиттів числа N.
ТУ. У вхідному потоці задано число N (N<=80). У вихідний потік вивести кількість розбиттів.
Вхідні дані
4
Вихідні дані
5
Для 4-х можна отримати такі розбиття: 4, 3+1, 2+2, 2+1+1, 1+1+1+1.

Задача 8-13
Реалізувати «Індійський алгоритм» піднесення до степеня який працює за таким правилом:

$$x^n = \begin{cases} x, & n=1 \\ x * (x^{n \text{ div } 2})^2, & n>1 \text{ і } n - \text{непарне} \\ (x^{n \text{ div } 2})^2, & n>1 \text{ і } n - \text{парне} \end{cases}$$

ТУ. У стандартному вхідному потоці дано цілі x та n (0<x<10, 0<=n<50). У вихідний потік вивести значення функції. Гарантується, що результат не буде перевищувати 10^{18}.
Вхідні дані
2 4
Вихідні дані
16

Задача 8-14

Дано N впорядкованих за зростанням натуральних чисел. Визначити чи є серед даних чисел число K.
ТУ. У першому рядку вхідного потоку дано числа N,K ($N<10^6$, $K<10^9$). Наступний рядок містить N чисел. У вихідний потік вивести Yes або No в залежності від наявності числа K.
Вхідні дані
5 101
2 5 10 101 103
Вихідні дані
Yes

Задача 8-15
Знайти значення функції Аккермана, що задається таким чином:

$$A(m, n) = \begin{cases} n+1, & m=0 \\ A(m-1,1), & m>0, n=0 \\ A(m-1, A(m, n-1)), & m>0, n>0 \end{cases}$$

ТУ. У вхідному потоці в одному рядку дано натуральні m, n ($m<5, n<1000$). Вхідні дані підібрані таким чином, що результат можна отримати. У вихідний потік вивести значення функції.
Вхідні дані
3 1

Вихідні дані
13

Розділ 9. Множини

Завдання для самоконтролю

Задача 9-1

Задано множини A та B, що складаються з двоцифрових чисел. Знайти суму тих елементів, що входять як у множину A, так і в множину B.

ТУ. У першому рядку знаходиться число N – кількість чисел множини A. У наступному рядку міститься N чисел. Дальше у новому рядку задається число M – кількість чисел множини B і у четвертому рядку містяться самі числа цієї множини. У вихідний потік вивести суму чисел.

Ввід<
3
10 12 13
4
11 12 13 14
Вивід>
25

Задача 9-2

Задано множини A та B, що складаються з двоцифрових чисел. Знайти суму тих елементів, що входять хоча би в одну з цих множин.

ТУ. У першому рядку знаходиться число N – кількість чисел множини A. У наступному рядку міститься N чисел. Дальше у новому рядку задається число M – кількість чисел множини B і у четвертому рядку містяться самі числа цієї множини. У вихідний потік вивести суму чисел.

Ввід<
3
10 12 13
4
11 12 13 14
Вивід>
60

Задача 9-3

Задано множини A та B, що складаються з двоцифрових чисел. Знайти різницю (A-B).

ТУ. У першому рядку знаходиться число N – кількість чисел множини A. У наступному рядку міститься N чисел. Дальше у новому рядку задається число M – кількість чисел множини B і у четвертому рядку містяться самі числа цієї множини. У єдиний рядок вихідного потоку вивести числа, що входять до (A-B).

Ввід<
4
11 12 13 14
3
10 12 13
Вивід>
11 14

Задача 9-4

Задано множини A, B та C, які складаються з цілих чисел не більших 255. Знайти числа, що входять хоча би в одну із множин A або B, але не входять у множину C.

ТУ. У першому рядку знаходиться число N – кількість чисел множини A. У наступному рядку міститься N чисел. Дальше у новому рядку задається число M – кількість чисел множини B і у четвертому рядку містяться самі числа цієї множини. В наступному рядку міститься K – кількість чисел множини C і потім у новому рядку самі елементи множини C. У єдиний рядок

вихідного потоку вивести числа, що входять мають властивості описані в умові задачі. Числа виводити у порядку зростання.
Ввід<
4
1 2 6 4
3
2 3 4
3
1 4 5
Вивід>
2 3 6

Задача 9-5
Дано два слова, що складаються з малих латинських літер. Вивести ті літери, що не входять в жодне із слів.
ТУ. В першому рядку знаходиться перше слово, у другому – друге. У вихідний потік вивести малі літери латинського алфавіту.
Ввід<
qwertyuiop
asdfghjkl
Вивід>
zxcvbnm

Задача 9-6
Заданий текст, що складається з малих латинських літер. Вивести ті літери, що зустрічаються в тексті не менше двох разів.
ТУ. У вхідному потоці знаходиться текст довжиною не більше 255 символів. Вивести в алфавітному порядку через пропуск літери, що зустрічаються більше двох разів.

Ввід<
the essence of
Вивід>
e s

Задача 9-7
Заданий текст, що містить малі літери латинського алфавіту і цифри. Вивести голосні, приголосні та цифри, що зустрічаються у даному тексті.
ТУ. У вхідному потоці знаходиться текст довжиною не більше 255 символів. У вихідний потік вивести у першому рядку голосні, у другому – приголосні, у третьому – цифри. Вивід символів здійснювати у порядку зростання їх кодів.

Ввід<
the essence of 2007
Вивід>
eo
cfhnst
027

Задача 9-8
Заданий текст із латинських літер, в кінці – крапка. Вивести на друк усі літери, які входять до тексту один раз. Великі та малі літери не розрізняти, при виведенні використовувати лише малі літери.

ТУ. У вхідному потоці дано текст, що закінчується крапкою. Вивести в алфавітному порядку літери, що зустрілися в тексті лише один раз.

Ввід<
Olimpiads.
Вивід>
adlmops

задача 9-9
Задана послідовність слів, розділених пропусками. Знайти кількість голосних в найдовшому слові. Великі та малі літери не розрізняти, при виведенні використовувати лише малі літери. Якщо таких слів є декілька, то слід брати перше у тексті.
ТУ. У вхідному потоці знаходиться текст із латинських літер довжиною не більше 255 символів. Вивести у вихідний потік число – кількість голосних у найдовшому слові.
Ввід<
the essence of
Вивід>
3

Задача 9-10
Задані n натуральних чисел. Для кожного введеного числа надрукувати в порядку зростання усі цифри, що не входять в десятковий запис цього числа.
ТУ. У першому рядку міститься число N (N<=1000). У наступних N рядках знаходяться десяткові записи чисел. Кількість цифр чисел не перевищує 100. Вивести у N рядках цифри, що відповідають умові задачі. Якщо таких цифр немає, то виводити порожній рядок.
Ввід<
1
16
Вивід>
02345789

Задача 9-11
Є група учнів, яких ми умовно будемо іменувати символами A..Z.
Також відомо хто входить в групи: Male, Female, Aerob, Karate. Знайти хто із учнів займається аеробікою та карате?
ТУ. В чотирьох різних рядках задано списки учнів, що належать даним групам у такому порядку: Male, Female, Aerob, Karate. У вихідний потік вивести по одному у рядку імена учнів, що займаються аеробікою та карате, причому спочатку вивести хлопців в алфавітному порядку, а потім дівчат.

Ввід<
A B R
C D E
A D E
B D A
Вивід>
A
D

Задача 9-12
Заданий рядок символів. Вивести символи, що не є буквами або цифрами.

ТУ. Задано текст довжиною не більше 30000 символів, що закінчується символом ‘#’. У тексті можуть зустрічатися лише букви латинського алфавіту та символи пунктуації, розділові знаки. У вихідний потік вивести у порядку зростання їх кодів символи, що відповідають умові задачі.
Ввід<
The day of an English pupil.#
Вивід>
.

Задача 9-13
Заданий текст із латинських літер, кінець якого позначений крапкою. Вивести перші входження букв до тексту, зберігаючи їх взаємний порядок.
ТУ. Довжина вхідного тексту не більша 1000 символів. У вихідний потік вивести літери в одному рядку. Розрізняти великі та малі літери.

Ввід<
The day of an English pupil.
Вивід>
ThedayofnEglispu

Задача 9-14
Задана послідовність слів. Вважаючи перше слово зразком, вибрати ті слова, які можуть бути отримані із зразка шляхом переставлення його літер.
ТУ. У вхідному потоці задано число N (N<=1000). Дальше у N рядках знаходиться по одному слову довжиною не більше 100 символів, слова складаються з малих латинських літер. У вихідний потік вивести шукані слова по одному у рядку.
Ввід<
4
abba
baab
bara
aabb
Вивід>
baab
aabb

Задача 9-15
Задано текст. Вивести літери латинського алфавіту, що зустрічаються в тексті менше, ніж К разів.
ТУ. У першому рядку вхідного потоку задається число К (К<100). У наступному рядку знаходиться текст довжиною не більше 30000 символів, що закінчується символом ‘#’. У вихідний потік вивести символи, що задовільняють умову задачі. Великі та мали літери не розрізняти, виводити лише малі літери в алфавітному порядку.

Ввід<
1
The day of an English pupil. #

Вивід>
bcjkmqrvwxz

Розділ 10. Робота з файлами даних

Для роботи з файлами використовують спеціальні типи даних, які називають потоками. Потік **ifstream** слугує для роботи з файлами у режимі читання. Потік **ofstream** слугує для роботи з файлами у режимі запису. Для роботи в режимі як запису так і читання слугує потік **fstream.** У програмах на С++ при роботі з текстовими файлами необхідно підключити бібліотеки (заголовкові файли) **iostream** та **fstream.**

Для того щоб записати данні у текстовий файл, необхідно:

- Описати змінну типу **ofstream.**
- Відкрити файл з допомогою функції **open**.
- Записати інформацію у файл.
- Обов'язково закрити файл.

Для читання даних із текстового файлу, необхідно:

- Описати змінну типу **ifstream.**
- Відкрити файл з допомогою функції **open**.
- Прочитати інформацію із файлу. При зчитуванні чергової порції інформації перевіряти чи не досягнуто кінця файлу.
- Обов'язково закрити файл.

Виведення у файловий потік

cout являє собою об'єкт типу **ofstream** (вихідний потік). Заголовковий файл **iostream** визначає вихідний потік **cout**. Аналогічно, заголовковий файл **fstream** визначає клас вихідного файлового потоку з ім'ям **ofstream.** Для запису на диск файлу потрібно оголосити об'єкт типу **ofstream,** ввказавши ім'я необхідного вихідного файлу як символьний рядок:

```
ofstream fo;
fo.open("filename.txt");
```

Якщо при оголошенні об'єкта типу **ofstream** вказати ім'я файлу, на диску буде створений новий файл із зазначеним ім'ям, або перезаписаний файл із таким же ім'ям, якщо він уже існує. У програмі 10.1 створюється об'єкт типу **ofstream** і потім використовується операція **<<** для виведення трьох рядків тексту у файл **output.txt**:

```
#include"stdafx.h"//Програма 10.1
#include<iostream>
#include<fstream>
using namespace std;
int main()
   {
      ofstream fout;  // опис змінної fout типу потік
      fout.open("output.txt"); // відкрили файл
      fout<<"Program na C++ "<<" klas-7"<<'\n';  // запис інформації
      fout<<"Informatika"<<'\n';// запис інформації
      fout<<"2010 year."<<'\n';// запис інформації
      fout.close(); // закрили файл
   }
```

Запустіть на виконання цю програму. Якщо ви працюєте в середовищі WINDOWS, після цього можете використати «Блокнот» для перегляду вмісту щойно створеного файлу **output.txt** (з папки, де знаходиться ваш проект):

Читання із вхідного файлового потоку

Уведення з файлу можна реалізувати, використовуючи об'єкти типу **ifstream** (клас вхідних файлових потоків). Як і при виведенні, слід просто створити об'єкт, передаючи йому, як параметр, необхідне ім'я файлу:

```
ifstream fi;
fi.open("filename.txt");
```

Після цього з файлу можна читати дані.

У наступній програмі відкривається файл **output.txt**, створений за допомогою програми 10.1 і з нього читаються дані:

```
#include<iostream.h>              //Програма 10.2
#include<fstream.h>
#inc!ude<conio.h>
int main()
{
ifstream fin("output.txt");
char one[64], two[64], three[64];
fin >> one;
fin » two;
fin >> three;
cout << one « endl;
cout « two « endl;
cout « three « endl;
getch() ;
return 0;
}
```

```
#include"stdafx.h"
#include<iostream>                              //Програма 10.1
#include<fstream>
using namespace std;
int main()
{
      ifstream fin("output.txt");
      char one[64], two[64], three[64];
      fin >> one;
      fin >> two;
      fin >> three;
      cout<<one<<'\n';
      cout<< two <<'\n';
      cout<< three<<'\n';
      return 0;
}
```

Якщо ви відкомпілюєте й запустите цю програму, то вона замість усього тексту відобразить тільки перші три слова: Prog na C++

Це відбувається тому, що, подібно до **cin**, вхідні файлові потоки використовують порожні символи (пропуск), щоб визначити, де закінчується одне значення й починається інше.

Читання цілого рядка файлового уведення

Вище ви ознайомились з функцією сіп. get line для читання цілого рядка із клавіатури. Об'єкти типу ifstream також можуть використовувати getline для читання рядка файлового уведення. Наступна програма використовує функцію getline для читання всіх трьох рядків файлу:

```
#include<iostream.h>              //Програма 10.3
#include<fstream.h>
#include<conio.h>
```

Завдання для самоконтролю.

Примітка. У зв'язку з тим, що система автоматизованої перевірки використовує для зчитування вхідних даних файл input.txt або консоль, то у всіх задачах дані треба читати з файлу input.txt. Вихідні дані у всіх завданнях треба відправляти лише на консоль.

Задача 10-1

Знайти різницю між найбільшим та найменшим числом.

ТУ. Числа задаються у вхідному файлі по одному числу у рядку. Кількість чисел не більше 100000, числа не перевищують по модулю10000.

Приклад вхідних та вихідних даних

Вхідні дані

3
6
1
11

Вихідні дані

10

Задача 10-2

Знайти найдовше слово.

ТУ. У вхідному файлі задані слова по одному у кожному рядку. Вивести найдовше слово що першим зустрілося у переліку. Довжина слів не перевищує 255 символів, кількість слів не більше 10000.

Приклад вхідних та вихідних даних.

Вхідні дані

abba
words
files

Вихідні дані

words

Задача 10-3

Знайти у тексті кількість літер 'a' (латинський алфавіт).

ТУ. У вхідному файлі задається вхідний текст довжиною до 100000 символів. У вихідний потік вивести кількість повторів 'a'.

Приклад вхідних та вихідних даних.

Вхідні дані

abba

Вихідні дані

2

Задача 10-4

Серед даних чисел вибрати лише парні.

ТУ. У вхідному файлі задано через пропуск числа не більші 30000. Кількість чисел не перевищує 50000. У вихідний потік вивести парні числа по одному у рядку.

Приклад вхідних та вихідних даних.

Вхідні дані

12 1001 12 53

Вихідні дані

12
12

Задача 10-5
Задано N натуральних чисел. Знайти число з найбільшою сумою цифр.
ТУ. У першому рядку вхідного файлу дано N (N<100000). У наступному рядку через пропуск задаються самі числа. У вихідний потік вивести перше з чисел з найбільшою сумою цифр.
Приклад вхідних та вихідних даних.
Вхідні дані
4
31 5 11 25
Вихідні дані
25

Задача 10-6
Задано N натуральних чисел. Знайти число з найбільшою сумою дільників.
ТУ. У першому рядку вхідного файлу дано N (N<1000). У наступному рядку через пропуск задаються самі числа. У вихідний потік вивести перше з чисел з найбільшою сумою дільників.
Приклад вхідних та вихідних даних.
Вхідні дані
4
31 5 11 25
Вихідні дані
31

Задача 10-7
Два прямокутники, сторони яких паралельні осям координат, задані координатами протилежних вершин. Знайти площу їх перетину.
ТУ. У вхідному файлі у двох різних рядках задані координати протилежних вершин прямокутників. Координати є цілими числами і по модулю не перевищують 10000. У вихідний потік вивести площу їх спільної частини. Якщо перетин є точка або пряма - вивести 0, якщо прямокутники не перетинаються - вивести -1.
Приклад вхідних та вихідних даних.
Вхідні дані
0 0 10 10
0 10 100 100
Вихідні дані
0

Задача 10-8
На площині задано N точок і координати двох протилежних вершин деякого прямокутника з сторонами паралельними осям координат. Вивести координати точок, що належать прямокутнику. Точки, що лежать на сторонах вважати такими, що належать прямокутнику.
ТУ. У першому рядку вхідного файлу знаходяться координати вершин прямокутника, у другому – число N (N<=100000) – кількість точок. У наступних N рядках міститься по два цілих числа в межах [-10000,10000] – координати точок. У вихідний потік вивести координати шуканих точок у порядку їх слідування у вхідному файлі.
Приклад вхідних та вихідних даних.
Вхідні дані
0 0 10 10
4
1 1
2 2
10 10

11 12
Вихідні дані
1 1
2 2
10 10

Задача 10-9
У проміжку [m,n] знайти просте число з найбільшою сумою цифр. Якщо таких чисел є декілька, то вивести те, що є найбільшим.
ТУ. У одному рядку вхідного файлу дано цілі числа m,n (0<m,n<30000). У вихідний потік вивести шукане число.
Приклад вхідних та вихідних даних.
Вхідні дані
1 10
Вихідні дані
7

Розділ 11. Структури даних

Структури. Структури й функції

З розділу 5 ви довідалися, що масив - це сукупність елементів одного типу і з'ясували, що використання масивів дуже зручне при написанні багатьох програм. Досить часто виникає потреба групувати зв'язану інформацію різних типів. Припустимо, що ваша програма працює з інформацією про школярів. Вона повинна відслідковувати дані про прізвище, вік, адресу, телефон, середній бал й т.д. Для зберігання цієї інформації програмі будуть потрібні змінні типу char, int, float, а також символьні рядки.

Для зберігання зв'язаної інформації різних типів у програмі використовують структури.

Оголошення структури

Структура визначає шаблон, за допомогою якого ваша програма може пізніше оголосити одну або декілька змінних. Інакше кажучи, у програмі слід спочатку визначити структуру, а потім можна оголошувати змінні типу цієї структури. Для визначення структури використовують ключове слово struct:

```
struct name   {
int name_l;  //  Оголошення елементів  структури
     float  name._2;
}  var;                                    //  Оголошення  змінної
```

У цьому прикладі визначена структура з ім'ям name, яка об'єднує два елементи: name__l типу int та name_2 типу float. Відразу ж оголошена змінна var типу name.

Для ініціалізації структури значення її елементів можна перерахувати у фігурних дужках у порядку їхнього опису:

```
struct {
char fio[30];
int date, code;
    double zarplata; } workers{"Petrov", 28, 118, 500.55};
```

Зверніть увагу, що в цьому випадку тип структури не має власного імені, тому не можна буде оголосити структуру такого ж типу в іншому місці програми.

Як і дані інших типів, структури можна об'єднувати в одно- та баготовимірні масиви. При ініціалізації масивів структур варто брати у фігурні дужки кожен елемент масиву (з огляду на те, що багатовимірний масив - це масив масивів):

```
struct coordinate{
int x, y;
}  coord[2][3]={
{{1,1},{3,2},{5,4}},   //масив  coord[0] {{8,0},{1,7},{1,0}},   //масив  coord[l] };
```

Наступний приклад демонструє визначення структури, що міститиме інформацію про учня:

```
struct uchen  { char name[64]   ; int  riknar; float  serball; char
phone[10]; };
```

Після визначення структури, у програмі також можна оголосити змінні типу цієї структури, як показано нижче:

uchen shkoiyar, star_shkolyar, new_shkolyar; У цьому випадку оператор створює три змінні типу структури uchen.

Використання елементів структури

Структура дозволяє групувати інформацію різних типів, що складає елементи (інша назва - поля) структури, в одній змінній. Щоб присвоїти значення елементу або звернутися до значення елемента, використовуватимемо операцію «вибір елемента» -крапка (.)■ Присвоїмо значення різним елементам змінної з ім'ям shkoiyar типу uchen:

```
strcpy(shkoiyar.name, "Ivanov Sergiy");
shkoiyar.riknar = 1998;
shkoiyar.serball=7.5;
strcpy(shkoiyar.phone, и25-56-89п);
```

Наступна програма ілюструє використання структури типу **uchen:**

```
#include<string.h>                //Програма 8.1
#include<iostream.h> #include<conio.h> ' int main()
```

```
{                                                    . ,
struct uchen {
char name[64] ;
int riknar;
float serball;
      char phone[10]; }shkolyar;
// Копіювати ім'я в поле-рядок strcpy(shkolyar.name, "Ivanov
Sergiy"); // Присвоювання значень числовим полям shkolyar.riknar =
1998; shkolyar.serball=7.5;
// Копіювати номеру телефону в поле-рядок s trcpy(shkolyar.phone, "25-56-89");
cout«"Uchen: "« shkolyar.name << endl; cout«"Rik nar : "<< shkolyar.riknar
<< endl; cout<<"Ser. bal.: "<< shkolyar.serball « endl; cout<<"Tel: "<<
shkolyar.phone << endl; getch(); return 0; }
```

Як бачите, надати значення елементам типів **int** та **float** дуже просто: використовується оператор присвоювання. Однак зверніть увагу, що для копіювання символьних рядків у елементи **name** й **phone** довелося скористатись функцією s**trcpy.**

*Отже, звертання до поля структури (наприклад, **shkolyar. name)** використовують так само, як і імена простих змінних такого ж типу.*

Структури й функції і

Якщо функція не змінює структуру, ви можете передати структуру у функцію за ім'ям. Наприклад, у програмі 8.2 функція **show_uchen** використовується для виведення елементів структури типу **uchen:**

```
#include<string.h>
#include<iostream.h> #include<conio.h> struct uchen
{
char name[64] ;
int riknar;.
float serball;
   char phone[10];; };
//функція виведення на екран void show_uchen (uchen. shkolyar) {
cout«"Uchen: "« shkolyar.name « endl; ■ cout«"Rik nar.: "« shkolyar.riknar
« endl; cout«"Ser. bal: "« shkolyar.serball « endl; cout«"Tel: "«
shkolyar.phone « endl;
}
int main()
{
uchen shkolyar;   //оголошення змінної-структури
//заповнення даними
strcpy(shkolyar.name, "Ivanov Sergiy");
shkolyar.riknar =1998;
shkolyar. serball=7.5;
strcpy(shkolyar.phone, "25-56-89");
//Звернення до функції виведення на екран
show uchen(shkolyar);
get'ch () ; return 0 ;
}  '
```

Зверніть увагу, що програма тепер визначає структуру **uchen** поза головною функцією **main** і до функції **show_uchen.** Оскільки функція оголошує змінну **shkolyar** типу **uchen,** визначення структури **uchen** має розташовуватися до функції.

Функції, що змінюють елементи структури

Як ви знаєте, якщо функція змінює параметр, його належить передавати у функцію за допомогою адреси. Це повністю стосується й структур: якщо функція змінює елемент структури, слід передавати цю структуру у функцію за допомогою адреси. Для передачі змінної типу структури за допомогою адреси перед ім'ям змінної ставлять знак операції взяття адреси (&): fun (fcshkolyar);

Всередині функції, що змінює один або кілька елементів, належить працювати з вказівником.

Наприклад, у наступній програмі структура типу uchen передається у функцію з ім'ям get_uchen_serball, що запитує в користувача середній бал учня й потім присвоює уведене число елементу структури serball. Щоб змінити елемент, функція Працює з вказівником на структуру:

```
#include<string.h>
#include<iostream.h>
#include<conio.h>
//Оголошення структури
struct uchen {
char name[64] ;
int riknar;
float serball;
char phone[10];
};
//функція виведення на екран void show_uchen(uchen shkolyar)
{
cout<<"Uchen: "«shkolyar.name«endl ;
cout«"Rik nar: "«shkolyar.riknar «endl;
cout«" Seredniy ball: " «shkolyar. serball«endl;
cout«" Tel: "« shkolyar. phone«endl ;
}
//функція уведення середнього балу
void get uchen serball(uchen *shkolyar)
{
cout<<"Uvedit seredniy ball:";
cin>>shkolyar->serball;
}
int main()
{
uchen shkolyar;  //оголошення змінної-структури
```

Завдання для самоконтролю

У всіх задачах вхідні дані треба читати з файлу **input.txt**, а виводити у стандартний вихідний потік.

Задача 11-1

Дано назви геометричних фігур та їх площу. Відсортувати вхідні дані у порядку зростання спочатку назви фігури, а потім її площі.

ТУ. У першому рядку вхідного файлу міститься ціле число N(N<1000) – кількість різних даних фігур. Дальше у наступних 2*N рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 20 символів і складається з малих латинських літер, а площа є дійсним числом з двома знаками після коми. У вихідний потік вивести впорядковані дані у такому форматі: в одному рядку назва фігури і через пропуск її площа.

Вхідні дані

3
kolo
2.52
figura
10.12
kolo
3.25

Вихідні дані

figura 10.12
kolo 2.52
kolo 3.25

Задача 11-2

Дано назви геометричних фігур та їх площу. Знайти фігуру із площею, що є найближчою до площі першої у списку фігури. Якщо таких є декілька, то вивести ту, що йде у переліку раніше.

ТУ. У першому рядку вхідного файлу міститься ціле число N(N<1000) – кількість рядків даних. Дальше у наступних 2*N рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 20 символів і складається з малих латинських літер, а площа є дійсним числом з двома знаками після коми. У вихідний потік вивести в одному рядку назву фігури і через пропуск її площу.

Вхідні дані

3
kolo
2.52
figura
10.12
kolo
3.25

Вихідні дані

kolo 3.25

Задача 11-3

Дано назви геометричних фігур та їх площу. Знайти фігури із найбільшими та найменшими площами.

ТУ. У першому рядку вхідного файлу міститься ціле число N(N<1000) – кількість рядків даних. Дальше у наступних 2*N рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 20 символів і складається з малих латинських літер, а площа є дійсним числом з двома знаками після коми. У вихідний потік виводити спочатку дані про

фігури з найбільшими площами, а потім з найменшими. Вивід здійснювати в окремому рядку для кожної фігури. Дані про жодну фігуру не можуть бути виведені більше одного разу.

Вхідні дані

3
kolo
2.52
figura
2.52
kolo
3.25

Вихідні дані

kolo 3.25
kolo 2.52
figura 2.52

Задача 11-4

Дано назви геометричних фігур та їх площу. Знайти сумарну площу для кожної назви фігури.

ТУ. У першому рядку вхідного файлу міститься ціле число N(N<1000) – кількість рядків даних. Дальше у наступних 2*N рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 20 символів і складається з малих латинських літер, а площа є дійсним числом з двома знаками після коми. У вихідний потік в окремих рядках виводити назву фігур та через пропуск їх загальну площу.

Вхідні дані

3
kolo
2.52
figura
2.52
kolo
3.25

Вихідні дані

kolo 5.77
figura 2.52

Задача 11-5

Дано назви геометричних фігур та їх площу. Які з фігур мають найбільшу загальну площу?

ТУ. У першому рядку вхідного файлу міститься ціле число N(N<1000) – кількість рядків даних. Дальше у наступних 2*N рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 20 символів і складається з малих латинських літер, а площа є дійсним числом з двома знаками після коми. У вихідний потік вивести назву фігур та через пропуск їх загальну площу. Якщо таких назв фігур є декілька, то виводити лише ту, що йде у переліку раніше.

Вхідні дані

3
kolo
2.52
figura
2.52
kolo
3.25

Вихідні дані

kolo 5.77

Задача 11-6

Дано список прізвищ N учнів з їх оцінками по K предметах. Знайти середні бали для кожного учня.

ТУ. У першому рядку вхідного файлу дано цілі числа N, K(0<N<=1000, 0<K<10). Дальше у 2*N рядках дані розміщені таким чином: спочатку в окремому рядку іде прізвище, а у наступному через пропуск знаходяться K оцінок. Прізвища довжиною не більше 10 символів. У вихідний потік вивести в окремих рядках прізвища учнів і їх середні бали.

Вхідні дані

3 4
Ivanov
4 4 5 5
Petrov
2 3 3 2
Sidorov
5 5 5 5

Вихідні дані

Ivanov 4.5
Petrov 2.5
Sidorov 5.0

Задача 11-7

Дано список прізвищ N учнів з їх оцінками по K предметах. Вивести список учнів, що мають максимальний середній бал.

ТУ. У першому рядку вхідного файлу дано цілі числа N, K(0<N<=1000, 0<K<10). Дальше у 2*N рядках дані розміщені таким чином: спочатку в окремому рядку іде прізвище, а у наступному через пропуск знаходяться K оцінок. Прізвища довжиною не більше 10 символів. У вихідний потік вивести в окремих рядках прізвища учнів і їх середні бали.

Вхідні дані

3 4
Ivanov
4 4 5 5
Petrov
2 3 3 2
Sidorov
5 5 5 5

Вихідні дані

Sidorov 5.0

Задача 11-8

Дано список M класів з прізвищами Ni учнів, що там навчаються, а також з їх оцінками по Ki предметах. Вивести список класів з найвищим середнім балом по всіх предметах для даного класу.

ТУ. У першому рядку вхідного файлу дано кількість класів M (0<M<100). У наступному рядку знаходиться назва класу з довжиною не більше 4-х символів. Після цього у новому рядку містяться цілі числа Ni, Ki (0<Ni<=40, 0<Ki<10). Дальше дані розміщені таким чином: спочатку в окремому рядку знаходиться прізвище, а у наступному через пропуск знаходяться K оцінок. Дальше все повторюється для наступного класу. Прізвища довжиною не більше 10 символів. У вихідний потік вивести в окремих рядках назви класів з найвищими середніми балами у порядку їх слідування у списку.

Вхідні дані

2
10-A

3 4
Ivanov
4 4 5 5
Petrov
2 3 3 2
Sidorov
5 5 5 5
10-B
2 3
Kukuschkin
4 4 4
Karasik
5 3 4
Вихідні дані
10-A 4.0
10-B 4.0

Задача 11-9
Дано список M класів з прізвищами Ni учнів, що там навчаються, а також з їх оцінками по K_i предметах. Визначити учнів з найвищим середнім балом.
ТУ. У першому рядку вхідного файлу дано кількість класів M (0<M<100). У наступному рядку знаходиться назва класу з довжиною не більше 4-х символів. Після цього у новому рядку містяться цілі числа Ni, Ki (0<Ni<=40, 0<Ki<10). Дальше дані розміщені таким чином: спочатку в окремому рядку знаходиться прізвище, а у наступному через пропуск знаходяться K оцінок. Дальше все повторюється для наступного класу. Прізвища довжиною не більше 10 символів. У вихідний потік вивести в окремих рядках вивести прізвище учня, його клас та середній бал.
Вхідні дані
2
10-A
3 4
Ivanov
4 4 5 5
Petrov
2 3 3 2
Sidorov
5 5 5 5
10-B
2 3
Kukuschkin
4 4 4
Karasik
5 3 4
Вихідні дані
Sidorov 10-A 5.0

Задача 11-10
Дано список M класів з прізвищами Ni учнів, що там навчаються, а також з їх оцінками по Ki предметах. Визначити кількість учнів з середнім балом більшим 4.
ТУ. У першому рядку вхідного файлу дано кількість класів M (0<M<100). У наступному рядку знаходиться назва класу з довжиною не більше 4-х символів. Після цього у новому рядку містяться цілі числа Ni, Ki (0<Ni<=40, 0<Ki<10). Дальше дані розміщені таким чином:

спочатку в окремому рядку знаходиться прізвище, а у наступному через пропуск знаходяться K оцінок. Дальше все повторюється для наступного класу. Прізвища довжиною не більше 10 символів. У вихідний потік вивести кількість учнів.

Вхідні дані

2
10-A
3 4
Ivanov
4 4 5 5
Petrov
2 3 3 2
Sidorov
5 5 5 5
10-B
2 3
Kukuschkin
4 4 4
Karasik
5 3 4

Вихідні дані

2

Задача 11-11

Прямокутники задані координатами лівої нижньої вершини та довжиною і висотою сторін. Визначити чи перетинаються хоча би два прямокутники. Прямокутники перетинаються тоді, коли площа їх перетину більша нуля.

ТУ. У першому рядку вхідного файлу дано ціле N (N<=1000) – кількість прямокутників. Дальше у наступних N рядках задаються Xi, Yi, Ai, Bi **(-10000<=Xi, Yi <=10000), (0<Ai,Bi<=10000)** відповідно координати вершини прямокутника, довжина і висота. У вихідний потік вивести “Yes” або “No”, що є відповіддю на поставлене запитання.

Вхідні дані

3
0 0 10 10
10 0 10 10
20 0 10 10

Вихідні дані

No

Задача 11-12

Прямокутники задані координатами лівої нижньої вершини та довжиною і висотою сторін. Знайти найбільшу кількість прямокутників, що перетинаються одночасно один з одним. Прямокутники перетинаються тоді, коли площа їх перетину більша нуля.

ТУ. У першому рядку вхідного файлу дано ціле N (N<=1000) – кількість прямокутників. Дальше у наступних N рядках задаються Xi, Yi, Ai, Bi (**-(-10000<=Xi, Yi <=10000), (0<Ai,Bi<=10000)** відповідно координати вершини прямокутника, довжина і висота. У вихідний потік вивести кількість прямокутників.

Вхідні дані

3
0 0 10 10
10 0 20 10
20 0 30 10

Вихідні дані

2

Задача 11-13

Прямокутники задані координатами лівої нижньої вершини та довжиною і висотою сторін. Знайти два прямокутники, площі яких відрізняються на найменшу величину.

ТУ. У першому рядку вхідного файлу дано ціле N (N<=1000) – кількість прямокутників. Дальше у наступних N рядках задаються Xi, Yi, Ai, Bi **(-10000<=Xi, Yi <=10000), (0<Ai,Bi<=10000)** відповідно координати вершини прямокутника, довжина і висота. У вихідний потік вивести в одному рядку два порядкових номери прямокутників, що задовольняють умову задачі. Перший номер має бути мінімально можливим і меншим за другий.

Вхідні дані

3
0 0 10 10
0 0 20 10
0 0 35 10

Вихідні дані

1 2

Задача 11-14

Прямокутники задані координатами лівої нижньої вершини та довжиною і висотою сторін. Знайти радіус найбільшого кола, яке може бути повністю поміщене в один із цих прямокутників.

ТУ. У першому рядку вхідного файлу дано ціле N (N<=1000) – кількість прямокутників. Дальше у наступних N рядках задаються Xi, Yi, Ai, Bi **(-10000<=Xi, Yi <=10000), (0<Ai,Bi<=10000)** відповідно координати вершини прямокутника, довжина і висота. У вихідний потік вивести радіус шуканого кола з точністю до десятих.

Вхідні дані

3
0 0 10 10
0 0 20 10
0 0 35 10

Вихідні дані

5.0

Задача 11-15

Прямокутники задані координатами лівої нижньої вершини та довжиною і висотою сторін. Знайти діагональ такого прямокутника, що вмістив би у собі всі дані прямокутники одночасно.

ТУ. У першому рядку вхідного файлу дано ціле N (N<=1000) – кількість прямокутників. Дальше у наступних N рядках задаються Xi, Yi, Ai, Bi **(-10000<=Xi, Yi <=10000), (0<Ai,Bi<=10000)** відповідно координати вершини прямокутника, довжина і висота. У вихідний потік вивести діагональ шуканого прямокутника з точністю до сотих.

Вхідні дані

3
0 0 10 10
10 0 20 10
20 0 30 10

Вихідні дані

50.99