

Дисципліна “Програмування”

Викладач Боярінова Юлія
Євгенівна

ib@ua.fm

067-175-13-08 (Київстар)

Методичне забезпечення

- Б.Л.Голуб \ Б.Л.Голуб, Є.М.Шукайло. Методичний посібник до вивчення дисципліни “Програмування та алгоритмічні мови”. Методичний посібник. – Видавничий центр НУБіП, 2012. – 64 с.
- Б.Л.Голуб \ Б.Л.Голуб, О.І.Примак. Методичні вказівки до виконання лабораторних робіт з дисциплін “Основи програмування”, “Програмування та алгоритмічні мови” (частина I). – Видавничий центр НУБіП, 2009. – 38 с.
- Б.Л.Голуб \ Б.Л.Голуб, О.І.Примак. Методичні вказівки до виконання лабораторних робіт з дисциплін “Основи програмування”, “Програмування та алгоритмічні мови” (частина II). – Видавничий центр НУБіП, 2009. – 50 с.

Рекомендована література

- **Базова**
- Б.Керниган \ Б.Керниган, Д.Ритчи. Язык программирования С. – Санкт-Петербург, 2001. – 300 с.
- Х.М.Дейтел \ Х.М.Дейтел, П.Дж.Дейтел. Как программировать на С. – М.:”Бином”, 2000. – 1005 с.
- **Додаткова**
- Крис Паппас \ Крис Паппас, Уильям Мюррей. Программирование на С и С++. Серия «Библиотека студента». – «Ирина», ВНУ, Киев, 2000. – 320с.
- ANSI, American National Standard for Information Systems – Programming Language C. – New York, 1990.

Лекція 1_Тема №1

Введення в програмування

1.1. Алгоритми та програми

- **Алгоритмом** називається визначена послідовність дій, виконання яких забезпечує досягнення кінцевої мети.
- **Алгоритмічна мова** – формальна система, призначена для запису алгоритмів.
- **Програма** – текст алгоритму, тобто це визначена послідовність дій, записаних на мові програмування, виконання яких призведе до кінцевої мети.
- **Програмування** – процес створення програми.

Структури алгоритмів

- 1) слідування (лінійна структура, коли перетворення інформації відбувається послідовно за певними формулами);
- 2) розгалуження (структури з перевіркою умов, коли перетворення інформації може здійснюватися за різними схемами, залежно від властивостей вхідних даних або проміжних результатів);
- 3) повторення (циклічні структури, коли є можливість багаторазового виконання деякої сукупності дій).

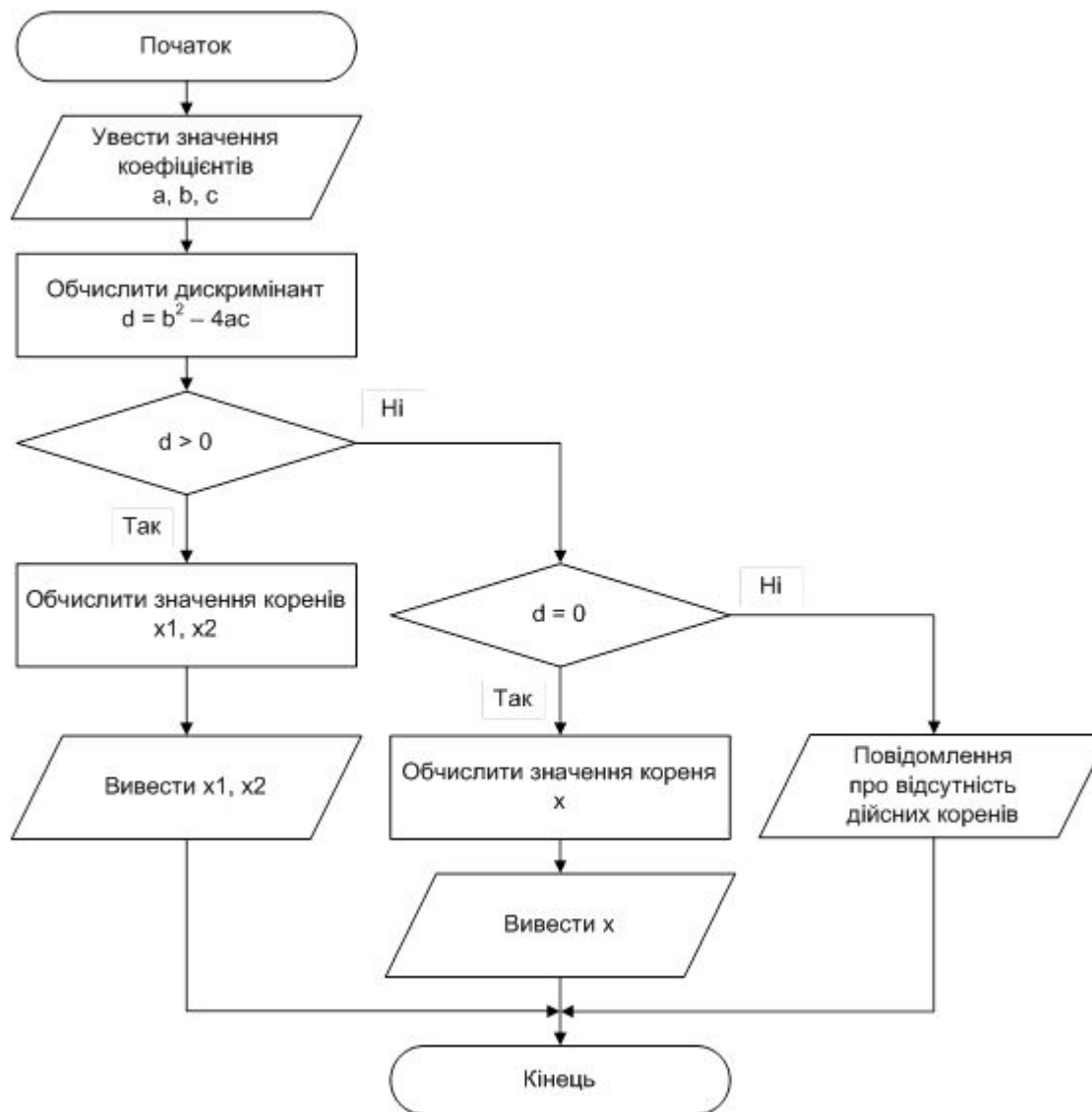
1.2. Блок-схема як засіб графічного зображення алгоритмів. Лінійні алгоритми. Алгоритми з розгалуженням. Алгоритми з циклами та циклічними структурами.

У словнику будь-якої мови знайдуться слова для опису трьох можливих структур алгоритму, для зображення алгоритму графічно розроблені спеціальні графічні фігури, в будь-якій мові програмування існують інструменти для реалізації лінійної структури, структури з перевіркою умов та циклічної структури. Способи опису алгоритмів: словесний; словесно-формульний; графічна схема; блок-схема; операторна схема; HIPO-схема; таблиця рішень, тощо.

В даному курсі ми будемо користуватись представленням алгоритмів у вигляді блок-схем.

Графічне зображення блоку	Найменування	Функція
	<p>Термінатор (пуск-зупинка)</p> <p>Дані (введення-виведення)</p>	<p>Позначення початку і кінця алгоритму.</p> <p>Перетворення даних у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Використовується для позначення будь-якої операції введення / виведення.</p>
	<p>Процес (операторний блок)</p>	<p>Виконання операції або групи операцій у результаті яких змінюється значення, форма представлення або розташування даних. Типове його використання – позначення оператора присвоювання.</p>
	<p>Рішення (умовний блок)</p>	<p>Вибір подальшого напрямку виконання алгоритму залежно від умови. Використовується для позначення умовного оператора або оператора варіанта.</p>
	<p>Зумовлений процес (підпрограма)</p>	<p>Відображає виконання процесу, який складається з однієї або кількох операцій, що визначені в іншому місці програми. Використовується для позначення виклику підпрограм (процедур і функцій).</p>
	<p>Межі циклу</p> <p>З'єднувач</p> <p>Коментар</p>	<p>Символ складається з двох частин - відповідно, початок і кінець циклу - операції, що виконуються всередині циклу, розміщуються між ними. Умови циклу і збільшення записуються всередині символу початку або кінця циклу - в залежності від типу організації циклу.</p> <p>Відображає зв'язок між перерваними лініями потоку інформації (наприклад, на іншій сторінці).</p> <p>Використовується для надання більш детальної інформації про кроки, процеси або групи процесів.</p>

Блок-схема алгоритму обчислення коренів квадратного рівняння



1.3. Мови програмування. Компілятори програмування

У вузькому сенсі під **програмуванням (кодуванням)** розуміють написання програм (інструкцій) на конкретній мові програмування. Відповідно, люди, які цим займаються, називаються **програмістами**.

Якщо висловлюватися мовою програміста, дії, які треба виконати – це **оператори**, а визначення послідовності їх виконання називається **керуванням програмою**.

Програмісти розробляють програми на різних мовах програмування, деякі з яких безпосередньо зрозумілі комп'ютеру, а інші поневіряються проміжній стадії – **трансляції**. Усі мови можуть бути поділені на три загальних типи:

- **машинні** мови;
- мови **асемблера**;
- мови **високого рівня**.

Компілятори програмування

Для перетворення програм на мові високого рівня у програми на машинній мові використовуються спеціальні програми, які називаються **компіляторами**.











Існують багато компіляторів та інтегрованих середовищ розробки:

- *Borland C++*
- *C++ Builder*
- *Microsoft Visual C++*
- *Microsoft Visual Studio*
- *Dev-C++*
- *Code::Blocks*
- *Embarcadero RAD Studio та ін.*

Основні мови програмування

- ***FORTRAN***
- ***COBOL***
- ***PASCAL***
- ***BASIC***
- ***C***
- ***C++***
- **Об'єктно-орієнтовні мови програмування**

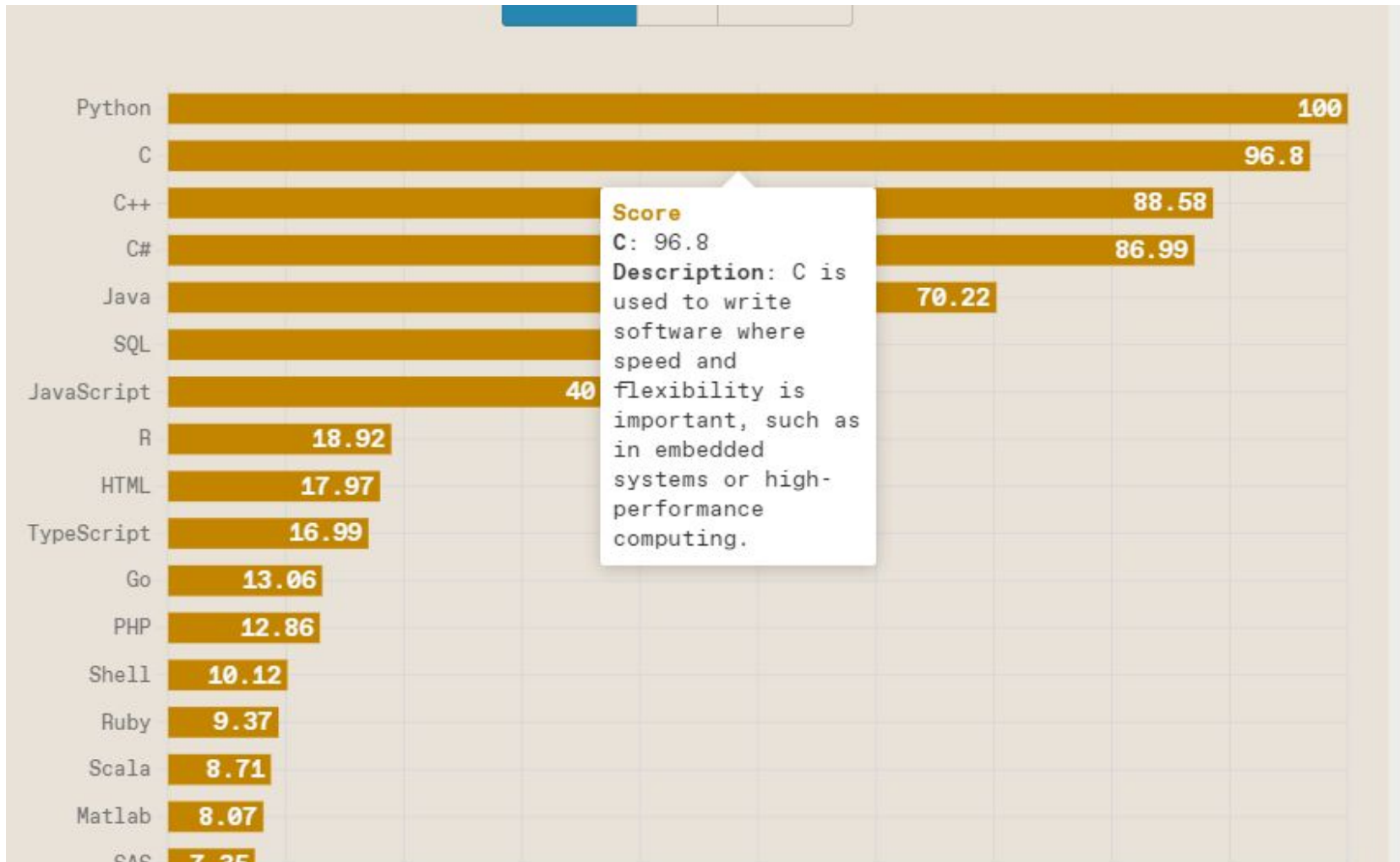
Рейтинг TIOBE Index

Sep 2022	Sep 2021	Change	Programming Language	Ratings	Change
1	2	▲	 Python	15.74%	+4.07%
2	1	▼	 C	13.96%	+2.13%
3	3		 Java	11.72%	+0.60%
4	4		 C++	9.76%	+2.63%
5	5		 C#	4.88%	-0.89%
6	6		 VB	4.39%	-0.22%
7	7		 JS	2.82%	+0.27%
8	8		 ASM	2.49%	+0.07%
9	10	▲	 SQL	2.01%	+0.21%
10	9	▼	 PHP	1.68%	-0.17%

Рейтинг TIOBE Index побудований на оцінці результатів пошукових запитів, що містять назву мови.

Логіка цього індексу дуже проста: «Якщо мову шукають в пошукових системах, то вона популярний». Звичайно ж, ця заява спірне, тому що програмісти-професіонали вкрай рідко будуть шукати в пошуковику саме назва мови програмування. Вони частіше шукають вирішення конкретного завдання. Але величезний плюс цього рейтингу в тому, що він досить об'єктивно показує інтерес до тій чи іншій мові.

Рейтинг IEEE Spectrum



- Щорічний рейтинг IEEE Spectrum Top Programming Languages використовує 11 метрик з 8-ми джерел, включаючи пошукові запити, згадки в твіттері і навіть згадки у вакансіях на роботу програміста.
- З одного боку цей рейтинг використовує більше даних, але з іншого боку в багатьох джерелах дані мають пов'язаний характер.
- Чим більше публікуються вакансій на деяку мову програмування, тим більше запитів буде в пошукових системах. Тобто у нових мов більше шансів потрапити на вершину рейтингу.

ІСТОРІЯ МОВИ С

ХАРАКТЕРИСТИКА С-СИСТЕМ

С - розроблена Деннісом Річі у 1972 році.

С++ - Б'єрном Страуструпом розроблено доповнення до мови С у 1983 році (мова С++ надає можливості об'єктно-орієнтованого програмування).

С# - об'єктно-орієнтована мова програмування. Розроблена в 1998-2001 роках групою інженерів під керівництвом Андерса Хейлсберга.

C-системи

C-системами - комплекс програмних продуктів, які дозволяють розроблювати програми на мові C.

Вони складаються з:

- середовища;
- мови програмування;
- стандартної бібліотеки.

Бібліотечні функції виконують такі операції, як введення/виведення, математичні обчислення.

Під час розробки C-програми проходять шість етапів: *редагування, передпроцесорна обробка, компіляція, компонування, завантаження, виконання.*

Алфавіт мови.

Алфавіт мови C/C++ 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 **командами** (інші назви — **оператори**). Команди будуються з лексем - неподільних елементів мови: слів, чисел, символів операцій. Слова поділяють на **ідентифікатори** і **ключові слова**. **Ідентифікатор** - це назва (ім'я), яку користувач надає об'єктам, наприклад, змінним, сталим, функціям. Усі слова можуть складатися з рядкових чи прописних літер англійського алфавіту, цифр, а також містити символ підкреслення. **Ідентифікатор завжди починається з букви або із символу підкреслення.** Однакові за змістом малі та великі літери у мові C++ вважаються різними символами. Зарезервовані ідентифікатори називаються **ключовими словами**. Вони використовуються для написання команд. **Змінити призначення ключового слова у програмі не можна.** **Основні ключові слова мови C/C++:** int, double, bool, main, new, break, printf, scanf, cin, cout, while, for, switch, struct, return і т.д.

Директиви препроцесора

Препроцесор - це програма, яка опрацьовує директиви.

Директиви препроцесора - це команди компілятора відповідної мови програмування, які виконуються на початку компіляції програми. **Директиви мови C/C++ починаються із символу #.**

- Директива **# include** означає, що до програми необхідно приєднати програмний код із зазначеного після неї файлу.
- Файли, які приєднують директивою **#include**, називаються файлами заголовків (header-файлами, бібліотеками).
- 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"**

Наприклад:

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

для підключення бібліотеки зі стандартними математичними функціями,

```
або #include "d:\stud\MyBib.h"
```

для підключення власного створеного бібліотечного файлу.

*Увага! Згідно зі стандартом ISO/ANSI файли заголовків для C++ у директиві **#include** в деяких середовищах програмування деякі бібліотеки прийнято записувати без розширення, наприклад **#include <iostream>**, **#include <math>**. Файли заголовків мови C, які використовуються у C++ - програмах, починаються з літери c, наприклад **#include <cstdio>**.*

Директива #define має подвійне значення.

По-перше, вона може задати стале значенню (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 (переозначити команди) чи записати функції:

```
#define D(a, b, c) ((b) * (b) - 4 * (a) * (c))
```

Тепер скрізь для обчислення дискримінанта замість команди $d=b*b-4*a*c$ можна записувати $d = D(a, b, c)$

Директива **#undef** скасовує дію директиви **#define**:

```
#define D(a,b,c) ((b) * (b) - 4 * (a) * (c))
```

```
#undef D
```

```
#define D(a,b,c) ((a) * (b) * (c))
```

Головна функція

Головна функція, яка має бути у кожній програмі - це функція вигляду:

```
int main(void)
{
тіло функції з командою return 0;
}
```

або

```
main()
{
тіло функції з командою return 0;
}
```

Ключове слово void означає, що функція не залежить від параметрів, його записувати не обов'язково.

Головна функція

або

```
void main()
```

```
{
```

```
тіло функції;
```

```
}
```

Така функція називається функцією `main()` типу `void`. Вона не повертає у програму жодних значень, тому команду `return` писати не треба.

Коментар

Коментар - це фрагмент тексту програми, який слугує для пояснення призначення програми чи окремих команд і не впливає на виконання команд. Його записують так:

//текст коментарю

або так:

/* текст коментарю */

У першому випадку коментар має бути або у кінці рядка, або єдиним у рядку. Другий спосіб більш універсальний: коментар можна записати будь-де, не розриваючи лексем.

Стандартний вигляд програми

```
// коментарі
```

```
#include <назва бібліотечного файла>
```

```
int main()
```

```
{
```

```
<тіло функції>;
```

```
return 0; // або return (0);
```

```
}
```

Приклад простої програми, виконання якої дозволить отримати на екрані монітора рядок тексту

```
/*Виведення рядка тексту*/  
#include <stdio.h>  
main()  
{  
    printf("\nЛаскаво запрошуємо на навчання в  
НУБіП!\n");  
return 0;  
}
```

Функція виведення даних printf

`printf ("Рядок Форматів", об'єкт 1, об'єкт 2, ..., об'єкт n);`

Рядок Форматів складається з наступних елементів:

- керуючих символів;
- тексту, представленого для безпосереднього виведення;
- форматів, призначених для виведення значень змінних різних типів.

Об'єкти можуть бути відсутніми.

Керуючі символи не виводяться на екран, а керують розташуванням символів, що виводяться. Відмінною рисою керуючого символу є наявність зворотного слеша '\ ' перед ним.

Текст виводиться у подвійних лапках.

Формати потрібні для того, щоб вказувати вид, в якому інформація буде виведена на екран. Відмінною рисою формату є наявність символу відсоток '%' перед ним.

Команди форматування (специфікатори формату) для printf ()

Код	Формат
%c	Символ типу char
%s	Рядок символів типу char
%d	Ціле число типу int зі знаком в десятковій системі числення;
%o	Ціле число типу int зі знаком в вісімковій системі числення
%x	Ціле число типу int зі знаком в шістнадцятковій системі числення
%u	Ціле число типу unsigned int;
%f	Десяткове число - формат одинарної точності з плаваючою точкою float
%lf	Десяткове число - формат подвійної точності з плаваючою точкою типу double
%ld	ціле число типу long int зі знаком в десятковій системі числення
%lu	Ціле число типу unsigned long int
%lx	Ціле число типу long int зі знаком в шістнадцятковій системі числення
%e	Десяткове число у вигляді x.x e + xx
%E	Десяткове число у вигляді x.x E + xx
%hd	Ціле число типу short зі знаком в десятковій системі числення
%hu	Ціле число типу unsigned short
%hx	Ціле число типу short зі знаком в шістнадцятковій системі числення

Основні керуючі символи

- '\ n' - новий рядок;
- '\ t' - горизонтальна табуляція;
- '\ v' - вертикальна табуляція;
- '\ b' - повернення на символ;
- '\ r' - повернення на початок рядка;
- '\ a' - звуковий сигнал.

Наприклад,

```
int main()
```

```
{
```

```
    printf ("Celebration:%c %d %s\n", 's', 22, " of  
September!");
```

```
    return 0;
```

```
}
```

призводить до результату

«Celebration: s 22 of September!».

Типи даних

Тип	Обсяг пам'яті	Діапазон
char	1 byte	Від -128 до 127
int	DOS,Win16- 2 bytes; Win32 – 4 bytes	Від -32768 до 32767 Від 2147483648 до 2147483647
unsigned int	DOS,Win16- 2 bytes; Win32 – 4 bytes	Від 0 до 65535
long int	DOS,Win16- 4 bytes; Win32 – 4 bytes	Від 2147483648 до 2147483647 Від -2^{63} до $(2^{63})-1$
float	4 bytes	Від $3.4 \cdot 10^{(-38)}$ до $3.4 \cdot 10^{(+38)}$

Типи даних(продовження)

Тип	Обсяг пам'яті	Діапазон
double	8 bytes	Від $1.7 \cdot 10^{-308}$ до $1.7 \cdot 10^{+308}$
long double	10 bytes	Від $3.4 \cdot 10^{-4932}$ до $3.4 \cdot 10^{+4932}$

Функція введення даних scanf()

Функція введення даних з клавіатури `scanf()` виконує читання даних, що вводяться з клавіатури, перетворює їх у внутрішній формат і передає функції. При цьому програміст задає правила інтерпретації вхідних даних за допомогою специфікацій рядка формату.

Загальна форма запису функції `scanf()`

```
scanf ("Рядок форматів", адреса змінної1, адреса змінної2, ...);
```

Рядок форматів аналогічний функції `printf()`.

Для формування адреси змінної використовується символ амперсанд '&':

Адреса = &об'єкт

Рядок форматів і список аргументів для функції обов'язкові.

Арифметика мови C

Дія в C	Арифметична операція	Алгебраїчний вираз	Вираз на C
Додавання	+	$a+7$	$a+7$
Віднімання	-	$p-3$	$p-3$
Множення	*	ap	$a*p$
Ділення	/	a/p	a/p
Обчислення залишка частки	%	$x \bmod y$	$x\%y$

Програма знаходить суму двох чисел

```
/* Програма знаходження суми */  
#include <stdio.h>  
main()  
{  
int d1, d2, sum; /* Об'ява змінних*/  
printf("Ввести перше число:"); /* Підказка */  
scanf("%d",&d1);/* Прочитати ціле */  
printf("Ввести друге число:"); /* Підказка */  
scanf("%d",&d2); /* Прочитати ціле */  
sum=d1+d2; /* Присвоїти суму */  
printf("Сума дорівнює %d",sum); /*Вивести суму */  
return(0); /* Програма закінчилася успішно */  
}
```

- Усі арифметичні операції є двомісними. Результат ділення двох цілих чисел також буде ціле число ($7/4=1$, $17/18=0$).
- Операцію обчислення залишка частки можна виконувати тільки з цілими числами.
- У мові C обчислення арифметичного виразу здійснюються за порядком, який відповідає правилам старшинства операцій

Операції присвоєння

У мові С передбачено декілька операцій присвоєння.

$c=c+3;$ або $c+=3;$

Будь-який оператор виду

змінна=змінна операція вираз;

може бути записаний у вигляді

змінна операція = вираз;

Бібліотека математичних операцій

<math.h>

sqrt(x) – знаходження кореня числа x

tan(x) – тангенс числа x

sin(x) – синус числа x

cos(x) – косинус числа x

log(x) – логарифм числа x

abs(x) – модуль числа x або ***fabs(x)***

pow(x,y) - піднесення числа x до степені y

Приклад

```
#include <stdio.h>
#include <math.h>
int main()
{
int a1=2, a2; /* Об'ява змінних*/
float b1,b2,rez;
a2=5;
printf("Ввести перше число:"); /* Підказка */
scanf("%f",&b1); /* Прочитати дійсне число */
printf("Ввести друге число:"); /* Підказка */
scanf("%f",&b2); /* Прочитати дійсне число */
rez=(b1*a2+a1*sin(b2))/log(b1-b2); /* Обчислення, якщо b1>b2*/
// rez=(b1*a2+a1*sin(b2))/log(abs(b1-b2));
printf("Результат дорівнює %f",rez); /*Вивести результат */
return 0; /* Програма закінчилася успішно */
}
```


Підключення української мови

1 спосіб	2 спосіб
<pre>#include <stdio.h> #include <locale.h> Int main() {int t; setlocale(LC_ALL,"Ukraine"); //char *locale = setlocale(LC_ALL, ""); printf ("Введіть значення\n");}</pre>	<pre>#include <stdio.h> #include <Windows.h> Int main() {int t; SetConcoleCP(1251); SetConcoleOutputCP(1251); printf ("Введіть значення\n");}</pre>

- **Windows-1251** (також вживаються назви **Win1251, CP1251**) — [кодування СИМВОЛІВ](#)) — кодування символів, що є стандартним 8-бітовим кодуванням для всіх локалізованих українських і російських версій [Microsoft Windows](#). Користується досить великою популярністю.

- **ASCII**(від назви *Американський стандартний код для інформаційного обміну*, англ. *American Standard Code for Information Interchange*) в обчислювальній техніці — система кодів, у якій числа від 0 до 127 включно поставлені у відповідність літерам, цифрам і символам пунктуації.
- ASCII (1963 рік) була розроблена для кодування символів, коди яких розміщувалися в 7 біт (128 символів; $2^7 = 128$); при цьому старший 7-й біт (нумерація з нуля) був використаний для контролю помилок, що виникли при передачі даних. Потім - кодування було розширено до 256 символів ($2^8 = 256$); коди перших 128 символів не змінилися. Такий 8-бітний варіант коду називають розширеним ASCII.
- Система широко використовується для зберігання тексту і передачі інформації між комп'ютерами.

Таблиця ASCII (кодикування Windows-1251)

0 -	16 - ►	32 -	48 - 0	64 - @	80 - P	96 - '	112 - p
1 - ☺	17 - ◀	33 - !	49 - 1	65 - A	81 - Q	97 - a	113 - q
2 - ☹	18 - ⇕	34 - "	50 - 2	66 - B	82 - R	98 - b	114 - r
3 - ♥	19 - !!	35 - #	51 - 3	67 - C	83 - S	99 - c	115 - s
4 - ♦	20 - ¶	36 - \$	52 - 4	68 - D	84 - T	100 - d	116 - t
5 - ♣	21 - §	37 - %	53 - 5	69 - E	85 - U	101 - e	117 - u
6 - ♠	22 - ▬	38 - &	54 - 6	70 - F	86 - V	102 - f	118 - v
7 -	23 - ⇕	39 - '	55 - 7	71 - G	87 - W	103 - g	119 - w
8 -	24 - ↑	40 - (56 - 8	72 - H	88 - X	104 - h	120 - x
9 -	25 - ↓	41 -)	57 - 9	73 - I	89 - Y	105 - i	121 - y
10 -	26 - →	42 - *	58 - :	74 - J	90 - Z	106 - j	122 - z
11 -	27 - ←	43 - +	59 - ;	75 - K	91 - [107 - k	123 - {
12 -	28 - ⊞	44 - ,	60 - <	76 - L	92 - \	108 - l	124 -
13 -	29 - ⇕	45 - -	61 - =	77 - M	93 - j	109 - m	125 - }
14 - 🎵	30 - ▲	46 - .	62 - >	78 - N	94 - ^	110 - n	126 - ~
15 - ⚙	31 - ▼	47 - /	63 - ?	79 - O	95 - ÷	111 - o	127 - ␣
16 - ►	32 -	48 - 0	64 - @	80 - P	96 -	112 - p	

128 -А	144 - Р	160 - а	176 - ☒	192 - L	208 - ⊥	224 - p	240 - Ě
129 - Б	145 - С	161 - б	177 - ☒	193 - ⊥	209 - ⊥	225 - с	241 - ë
130 - В	146 - Т	162 - в	178 - ■	194 - ⊥	210 - ⊥	226 - т	242 - ё
131 - Г	147 - У	163 - г	179 -	195 - ⊥	211 - ⊥	227 - у	243 - е
132 - Д	148 - Ф	164 - д	180 -]	196 - —	212 - ⊥	228 - ф	244 - ě
133 - Е	149 - Х	165 - е	181 -	197 - ⊥	213 - ⊥	229 - х	245 - ě
134 - Ж	150 - Ц	166 - ж	182 -	198 - ⊥	214 - ⊥	230 - ц	246 - ŷ
135 - З	151 - Ч	167 - з	183 - ⊥	199 - ⊥	215 - ⊥	231 - ч	247 - ŷ
136 - И	152 - Ш	168 - и	184 - ⊥	200 - ⊥	216 - ⊥	232 - ш	248 - °
137 - Й	153 - Щ	169 - й	185 - ⊥	201 - ⊥	217 - ⊥	233 - щ	249 - ●
138 - К	154 - Ъ	170 - к	186 - ⊥	202 - ⊥	218 - ⊥	234 - ъ	250 - .
139 - Л	155 - Ы	171 - л	187 - ⊥	203 - ⊥	219 - ⊥	235 - ы	251 - √
140 - М	156 - Ь	172 - м	188 - ⊥	204 - ⊥	220 - ⊥	236 - ь	252 - №
141 - Н	157 - Э	173 - н	189 - ⊥	205 - =	221 - ⊥	237 - э	253 - Ø
142 - О	158 - Ю	174 - о	190 - ⊥	206 - ⊥	222 - ⊥	238 - ю	254 - ■
143 - П	159 - Я	175 - п	191 - ⊥	207 - ⊥	223 - ⊥	239 - я	255 -
144 - Р	160 - а	176 - ☒	192 - L	208 - ⊥	224 - p	240 - Ě	

Приклад 1

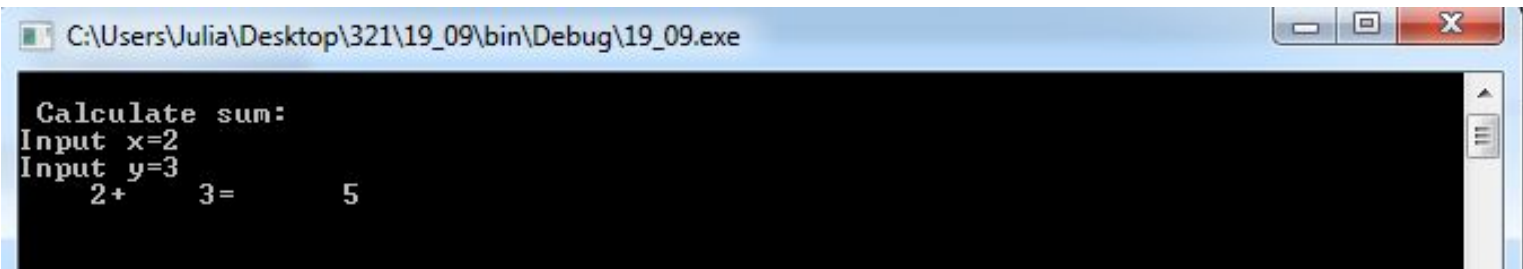
- Написати програму, яка обчислює суму двох цілих чисел

```
#include <stdio.h>
int main()
{int x,y;
printf("\n Calculate sum: ");
printf("\nInput x=");
scanf("%d",&x);
printf("\nInput y=");
scanf("%d",&y);
printf("%d+%d=%d",x,y,x+y);
return 0;
}
```

```
C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe  
Calculate sum:  
Input x=3  
Input y=4  
3+4=?_
```



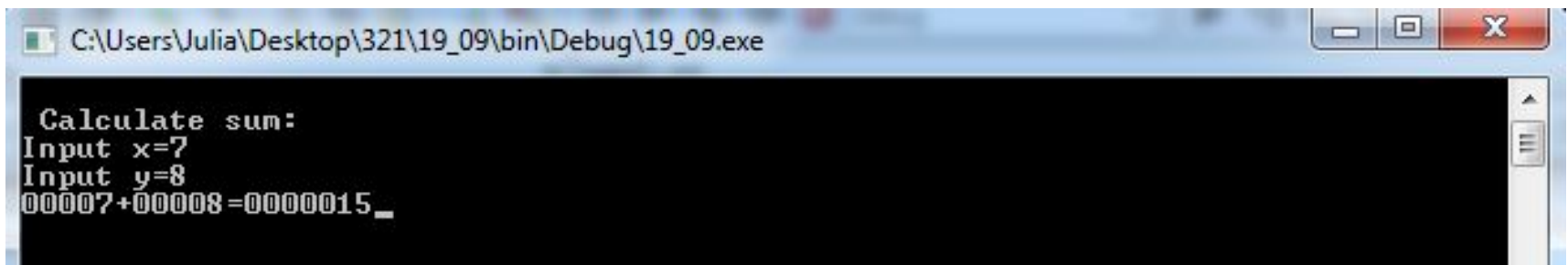
```
#include <stdio.h>
int main()
{int x,y;
printf("\n Calculate sum: ");
printf("\nInput x=");
scanf("%d",&x);
printf("\nInput y=");
scanf("%d",&y);
printf("%5d+%5d=%7d",x,y,x+y);
return 0;
}
```



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe. The window contains the following text:

```
Calculate sum:  
Input x=2  
Input y=3  
2+ 3= 5
```

```
printf("%05d+%05d=%07d",x,y,x+y);
```



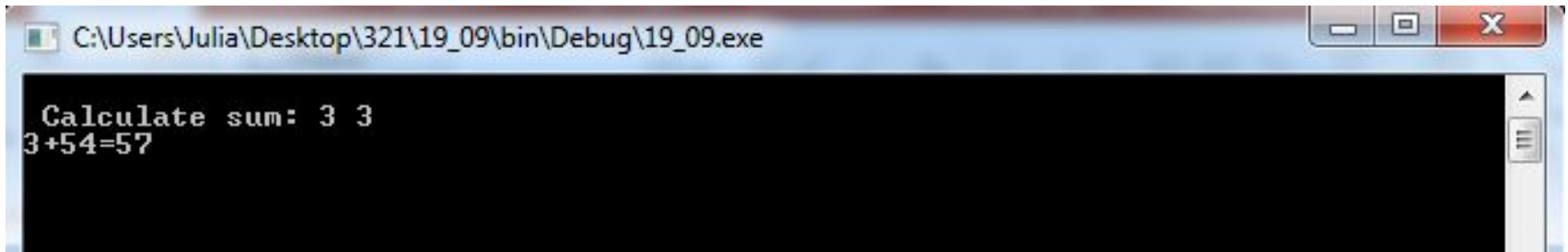
```
C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe  
Calculate sum:  
Input x=7  
Input y=8  
00007+00008=0000015_
```

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int x,y;
printf("\n Calculate sum: ");
scanf("%d%d",&x,&y);
printf("%d+%d=%d",x,y,x+y);
return 0;
}
```

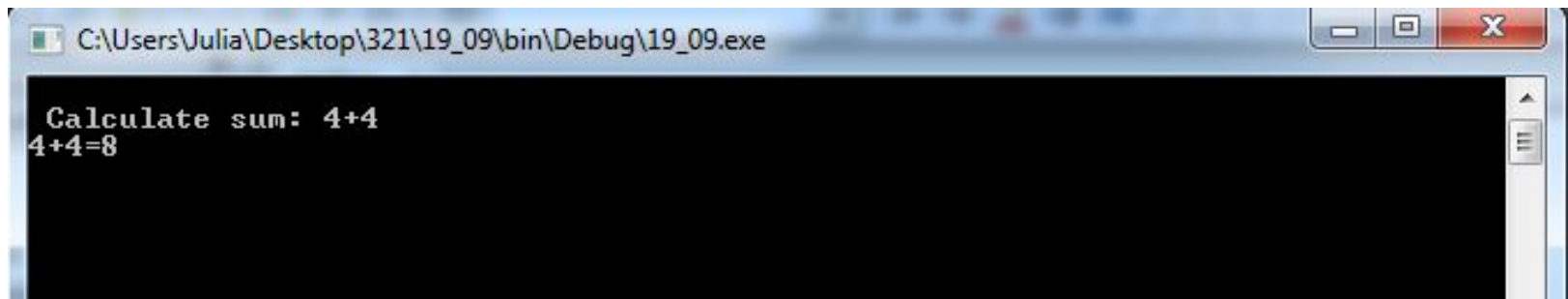
```
C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe  
Calculate sum: 4+5  
4+5=9_
```

```
C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe  
Calculate sum: 5 5  
5+5=10
```

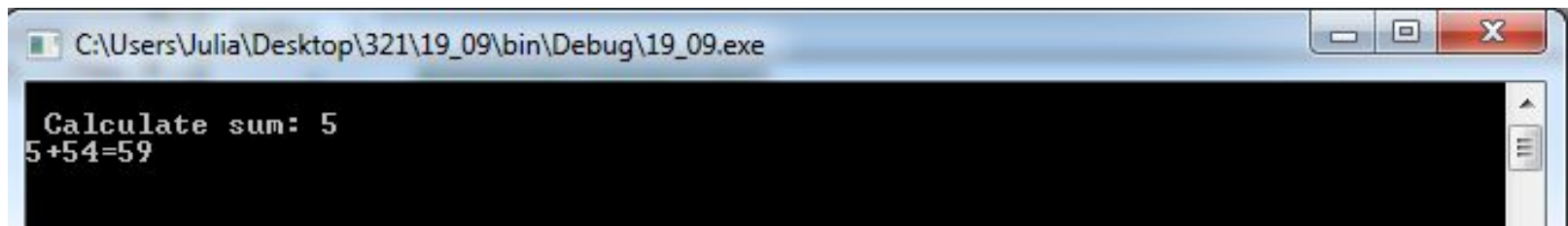
```
#include <stdio.h>
#include <stdlib.h>
int main()
{ int x,y;
printf("\n Calculate sum: ");
scanf("%d+%d",&x,&y);
printf("%d+%d=%d",x,y,x+y);
return 0;
}
```



```
C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe  
Calculate sum: 3 3  
3+54=57
```



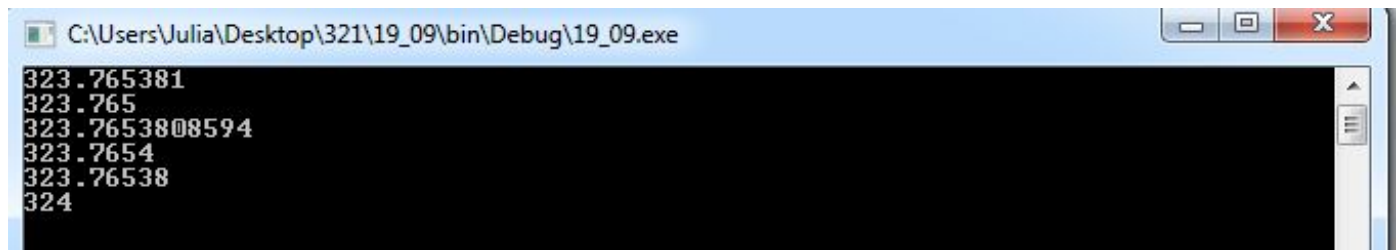
```
C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe  
Calculate sum: 4+4  
4+4=8
```



```
C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe  
Calculate sum: 5  
5+54=59
```

Формат виведення дійсних чисел

```
float b=323.765381;  
printf("%f\n",b);  
printf("%5.3f\n",b);  
printf("%3.10f\n",b);  
printf("%.4f\n",b);  
printf("%.5f\n",b);  
printf("%3.0f\n",b);
```



```
C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe  
323.765381  
323.765  
323.7653808594  
323.7654  
323.76538  
324
```


Приклад 2

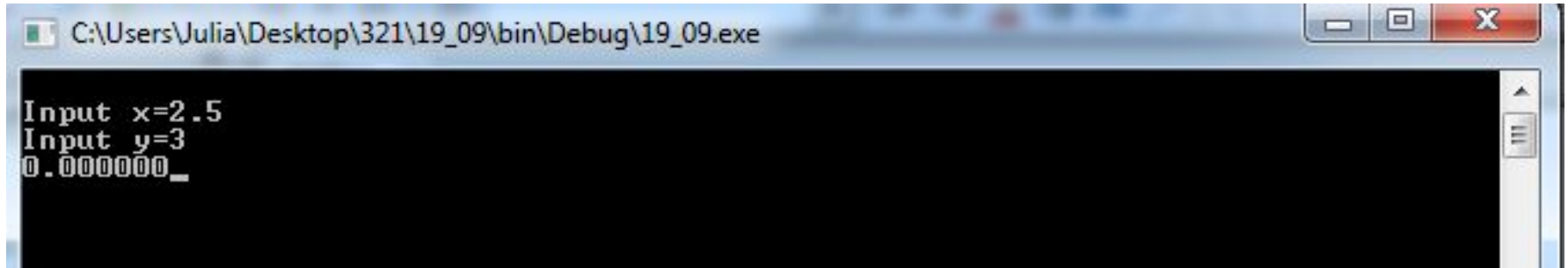
Обчислити вираз за формулою

$$t1 = \frac{1}{a}(x^2 + 2)(5y - \ln x)$$

Математичний вираз	Вираз мовою C
$t1 = \frac{1}{a}(x^2 + 2)(5y - \ln x)$	t1=1/a*(x*x+2)*(5*y-log(x)) або t1=1/a*(pow(x,2)+2)*(5*y-log(x))

```
#include <stdio.h>
#include <math.h>
int main()
{int a=2;
float x,y,t1;
printf("\nInput x=");
scanf("%f",&x);
printf("Input y=");
scanf("%f",&y);
t1=1/a*(x*x+2)*(5*y-log(x));
```

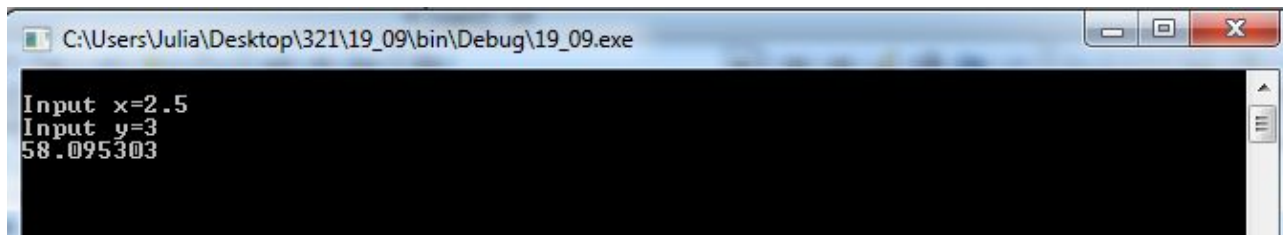
```
printf("%f",t1);  
return 0;  
}
```



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe. The window contains the following text:

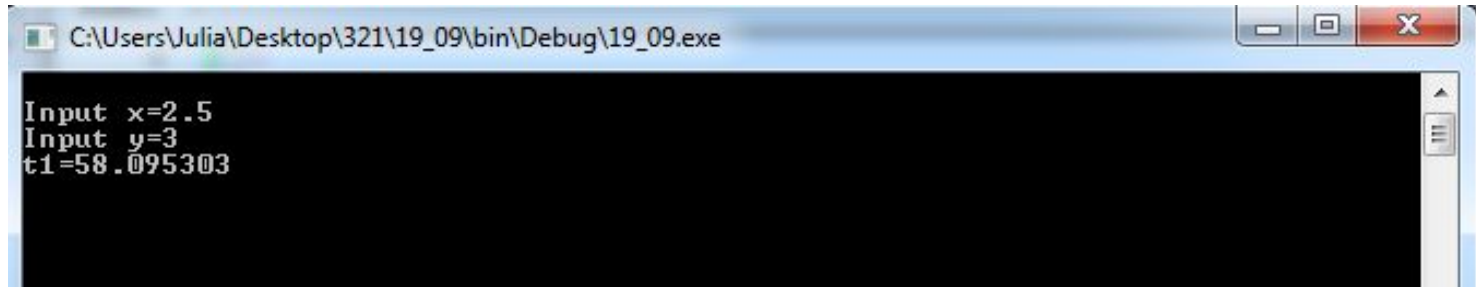
```
Input x=2.5  
Input y=3  
0.000000_
```

$t1=1.0/a*(x*x+2)*(5*y-\log(x));$



```
C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe
Input x=2.5
Input y=3
58.095303
```

`t1=1.0/a*(pow(x,2)+2)*(5*y-log(x));`



```
C:\Users\Julia\Desktop\321\19_09\bin\Debug\19_09.exe
Input x=2.5
Input y=3
t1=58.095303
```