

Передмова	6
Вступ	7
Історія розвитку програмування	8
I. МОВА ПРОГРАМУВАННЯ Cі	16
1. Базові поняття мови Cі	16
1.1. Алфавіт мови Cі	16
1.2. Ідентифікатори	16
1.3. Службові (зарезервовані) слова	16
1.4. Типи даних мови Cі	17
1.5. Константи і змінні	17
1.6. Дані цілого типу	18
1.7. Дійсні типи даних	19
1.8. Символьний тип даних	19
1.9. «Порожній» тип void	20
1.10. Сумісність типів	20
2. Структура простої програми мовою Cі	20
2.1. Директиви препроцесора Cі	22
2.1.1. Директива #include	22
2.1.2. Директива #define	22
2.1.3. Директива #undef	23
2.2. Коментарі	23
2.3. Операції в мові Cі	24
2.3.1. Основні операції мови Cі	24
2.3.2. Арифметичні операції	26
2.3.3. Операції поразрядної арифметики	26
2.3.4. Логічні операції	26
2.3.5. Операції відношення	27
2.3.6. Операції присвоєння	27
2.3.7. Умовна(тернарна) операція	27
2.3.8. Операція sizeof	27
2.3.9. Скорочені операції мови Cі	28
2.3.10. Вирази. Пріоритет операцій	28
3. Функції вводу-виводу в Cі	29
3.1. Функція форматного виводу printf()	30
3.2. Функція puts()	30
3.3. Функція putchar()	30
3.4. Функція форматного вводу scanf()	31
3.5. Функція gets()	31

3.6.	Функція <code>getchar()</code>	31
4.	Оператори мови Cі	31
4.1	Умовні оператори	32
4.1.1.	Оператор <code>if</code>	32
4.1.2.	Оператор вибору <code>switch</code>	32
4.2.	Оператори циклу	35
4.2.1.	Оператор <code>while</code>	35
4.2.2.	Оператор <code>do...while</code>	36
4.2.3.	Оператор <code>for</code>	36
4.2.4.	Вибір оператора циклу	37
4.3.	Оператори переходів <code>break</code> , <code>continue</code> , <code>return</code> , <code>goto</code>	38
4.4.	Порожній оператор	39
4.5.	Оператор вираз	40
5.	Масиви і вказівники в Cі	40
5.1.	Одновимірні масиви	40
5.2	Багатовимірні масиви	42
5.3	Вказівники	43
5.4	Рядки символів	44
6.	Функції користувача в Cі	45
6.1	Функція користувача і її прототип	45
6.2.	Функції із змінною кількістю аргументів	47
7.	Рекурсія	48
7.1.	Рекурсивні об'єкти	48
7.2.	Рекурсивні процедури і функції	49
7.3.	Нескінченна рекурсія	49
7.4.	Коли рекурсія не потрібна	50
7.5.	Рекурсивний пошук	51
7.6.	Перебір варіантів	51
7.6.1	Комбінації	52
7.6.2	Перестановки	52
8.	Структури в Cі	52
8.1	Опис шаблонів структур	53
8.2	Опис структур-змінних	54
8.3	Доступ до компонент структури	55
8.4	Анонімний опис структури і оператор задання типу <code>typedef</code>	56
9.	Робота з пам'яттю в Cі	56
9.1	Статична пам'ять	58
9.2	Стекова або локальна пам'ять	57

9.3	Динамічна пам'ять, або купа	58
10.	Робота з файлами	60
10.1	Файловий ввід/вивід	60
10.2	Відкриття файлу: функція <code>foopen()</code>	60
10.3	Діагностика помилок: функція <code>perrog()</code>	61
10.4	Закриття файлу: функція <code>fclose()</code>	62
10.5	Файловий ввід/вивід в текстовому режимі	62
10.5.1.	Поняття потоку вводу або виводу	64
10.5.2	Функції <code>scanf()</code> і <code>printf()</code> вводу/виводу в стандартні потоки	66
10.5.3	Функції текстового перетворення <code>sscanf()</code> і <code>sprintf()</code>	66
10.6	Файловий ввід/вивід в бінарному режимі	67
10.6.1	Функції бінарного читання і запису <code>fread()</code> і <code>fwrite()</code>	67
10.6.2	Приклад: підрахунок кількості символів і рядків у текстовому файлі	68
10.7	Інші корисні функції вводу-виводу	69
10.8	Низкорівневий ввід/вивід	73
10.8.1	Відкриття, створення, закриття і видалення	73
10.8.2	Довільний доступ – <code>lseek()</code>	73
II. ОСНОВИ АЛГОРИТМІЗАЦІЇ 74		
II.1.	Поняття алгоритму і його властивості	74
II.2.	Основні етапи алгоритмізації	75
II.3.	Правила оформлення блок-схем алгоритмів	76
II.4.	Використання синтаксичних діаграм Нассі-Шнейдермана для запису алгоритмів	79
II.5.	Використання псевдокоду для запису алгоритмів	81
II.6.	Задачі з розробки алгоритмів	82
II.6.1	Алгоритм обчислення скінченної суми	83
II.6.2.	Алгоритм розв'язання нелінійного рівняння методом дотичних (Ньютона)	83
II.6.3.	Алгоритм розв'язання нелінійного рівняння поділом відрізка навпіл	84
II.6.4	Алгоритми обробки одновимірних масивів	85
Максимальний елемент масиву x і його індекс (85). Мінімальний елемент масиву x і його індекс (85).		
II.6.5	Алгоритми обробки двовимірних масивів	90
Максимальний елемент матриці і його індекс (85). Мінімальний елемент матриці і його індекс (86). Транспонування матриці (86). Додавання матриць (86).		
II.6.6	Алгоритми пошуку	87
Послідовний пошук (87). Бінарний пошук (87).		
II.6.7	Основні алгоритми сортування масивів	87
Сортування методом вибору (87). Сортування вставками (88). Бульбашкове сортування (88). Сортування перемішуванням (89). Швидке сортування (89).		
Додаток 1. Позиційні системи числення і їх використання в обчислювальній техніці 90		
Додаток 2. Виконання програм мовою Сі в середовищі Geany 92		

Передмова

Дисципліна «Програмування мовою Сі» є базовою для багатьох курсів, що вивчаються студентами наступні чотири роки навчання. Базуючись на шкільних знаннях математики і фізики, вона в свою чергу служить фундаментом для вивчення інженерії апаратного та програмного забезпечення.

Під час вивчення курсу студенти знайомляться з основними принципами створення і візуалізації алгоритмів, парадигмою структурного програмування і реалізують її універсальною мовою програмування. Практичні навички, вироблювані курсом, містять алгоритмізацію і програмування інженерних завдань, аналіз, налагодження і тестування програмного коду.

Очевидно, що вибір першої мови програмування в значній мірі визначає стиль мислення початківця програміста, закладає основи підходів, в рамках яких надалі належить формувати навички та вміння, пов'язані з проблемно-орієнтованими мовами програмування, включно з високорівневими об'єкними підходами та моделями. Якою ж має бути перша мова програмування для вивчення майбутніми розробниками системного і прикладного програмного забезпечення?

Сі — це мова, створена свого часу для написання ОС Unix і доступна нині практично для будь-якої платформи, від мікроконтролерів до потужних обчислювальних кластерів. Сі на сьогоднішній день — основний засіб створення ядер операційних систем, драйверів пристроїв і просто високопродуктивного програмного коду. Крім того, синтаксис Сі, завдяки своїй структурованості і лаконічності ліг в основу безлічі високорівневих мов програмування, як загального призначення так і спеціальних: C++, C#, Java, PHP — лише кілька найбільш популярних прикладів.

Сі елегантна в синтаксисі, і дає змогу створювати дуже ефективні програми, які легко читаються і водночас легко переносяться на інші апаратні платформи. Однак ця мова створювалась для професійних системних програмістів, і тому є нетерпимою до неакуратного коду. Зворотню стороною високої ефективності є велика свобода, надана розробнику, на плечі якого лягає вся відповідальність за прийняті під час розробки програми рішення.

На відміну від мов, що спочатку створювалися як навчальні, Сі майже не контролює програміста. Рамки високорівневої мови програмування в Сі ілюзорні, їх легко обійти — через незнання або внаслідок прийнятих рішень. Концепції типів даних чітко окреслені, але при цьому зовсім прозорі, і крізь них проглядає апаратна архітектура, в якій все зводиться до машинних слів: послідовностей бітів фіксованої довжини, що задають дані або адреси комірок пам'яті. Сі має зручні та ефективні засоби роботи з векторними даними, такими як масиви і рядки, але самі абстракції цих типів зведені до режимів адресації, близьких за своєю суттю до реалізованих в апаратурі.

Сі довіряє програмісту, виконуючи будь які перетворення над типами або нестандартні поєднання синтаксичних конструкцій. Так можна створювати лаконічний і ефективний код — але так само легко зробити помилку, що приводить до катастрофічних результатів на етапі виконання програми. Перекладання відповідальності за дії програми на плечі програміста — крок неминучий у разі розробки системних програм, драйверів пристроїв і програм для високопродуктивних обчислень. Насправді цим і забезпечується швидкість написаного мовою Сі-коду.

Очевидно, що починати вивчати таку мову програмування краще на дуже простих програмах — наприклад таких, під час створення яких вивчаються основи алгоритмізації та структурного підходу. Практика показує, що програміст-початківець швидко засвоює як небезпечно допускати недбалість під час написання коду, виробляє необхідні навички і досить швидко починає створювати досить працездатний код.

Згодом, під час вивчення засобів створення складних програмних систем (наприклад, в рамках об'єктно-орієнтованого підходу) вироблена звичка до акуратного і вдумливого написання тексту програми повинна позитивно позначитися на якості створюваних програмних текстів і в підсумку привести до виходу на ринок праці висококваліфікованих інженерів-розробників електронно-інформаційних систем. Тому можна з упевненістю сказати, що мова Сі є прекрасним вибором для початкуючого системного програміста.