

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ПРОГРАМУВАННЯ – 1.

ПРОЦЕДУРНЕ ПРОГРАМУВАННЯ:

КОМП'ЮТЕРНИЙ ПРАКТИКУМ

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою
«Автоматизація та комп'ютерно інтегровані технології кібер-енергетичних систем»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»*

Київ
КПІ ім. Ігоря Сікорського
2021

Програмування – 1. Процедурне програмування (комп'ютерний практикум): [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / Укладачі: Ю. Є. Грудзинський, К. В. Крилов, П. В. Новіков, А. В. Сагун; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 4,8 МБайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 102 с.

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 8 від 24.06.2021 р.)
за поданням Вченої ради Теплоенергетичного факультету (протокол № 12 від 31.05.2021 р.)*

Електронне мережне навчальне видання

ПРОГРАМУВАННЯ – 1. ПРОЦЕДУРНЕ ПРОГРАМУВАННЯ. (комп'ютерний практикум)

Укладачі: *Грудзинський Юліан Євгенович, старший викладач
Крилов Костянтин Володимирович, асистент
Новіков Павло Валерійович, канд. техн. наук, доцент
Сагун Андрій Вікторович, канд. техн. наук, доцент*

Рецензент: *Сегеда Ірина Василівна, канд. екон. наук, доцент*

Відповідальний редактор: *Степанець Олександр Васильович, канд. техн. наук, доцент*

У посібнику розглядаються необхідні теоретичні відомості, наведені завдання до лабораторних робіт, які необхідні при вивченні дисципліни "Програмування – 1. Процедурне програмування".

Посібник призначений для студентів, які навчаються за спеціальністю 151 "Автоматизація та комп'ютерно інтегровані технології", освітньо-професійна програма "Автоматизація та комп'ютерно-інтегровані технології кібер-енергетичних систем".

Зміст

ВИМОГИ ДО ВИКОНАННЯ, ОФОРМЛЕННЯ І ЗДАЧІ	
КОМП'ЮТЕРНИХ ПРАКТИКУМІВ	5
1. Оформлення	5
2. Нарахування балів	6
3. Самостійне виконання завдань	6
4. Захист завдань	6
5. Угоди про стиль кодування	6
6. Структура проектів	6
7. Створення консольних програм в <i>MS Visual Studio</i> (на прикладі версії 2008).....	7
8. Рекомендації по створенню власних проектів в середовищі <i>Visual Studio</i>	10
Комп'ютерний практикум № 1. УМОВНІ ОПЕРАТОРИ МОВИ C ..	13
1.1. Теоретичні відомості.....	13
Умовний оператор <i>if</i>	13
Оператор-перемикач <i>switch</i>	15
1.2. Завдання на комп'ютерний практикум	17
1.3. Варіанти завдань	17
1.4. Зміст звіту.....	21
1.5. Контрольні питання.....	22
Комп'ютерний практикум № 2. ОПЕРАТОРИ ЦИКЛІВ МОВИ C ...	23
2.1. Теоретичні відомості	23
Оператор циклу <i>while</i>	23
Оператор циклу <i>for</i>	24
Оператор циклу <i>do while</i>	25
Програмування вкладених циклів	26
2.2. Завдання на комп'ютерний практикум	26
2.3. Варіанти завдань.....	27
2.4. Зміст звіту	31
2.5. Контрольні питання	32
Комп'ютерний практикум № 3. МАСИВИ.....	33
3.1. Теоретичні відомості	33
3.2. Завдання на комп'ютерний практикум.....	35
3.3. Варіанти завдань	36
3.4. Зміст звіту	42
3.5. Контрольні питання	42
Комп'ютерний практикум № 4. РОБОТА З РЯДКАМИ В МОВІ C	43
4.1. Теоретичні відомості	43
4.2. Завдання на комп'ютерний практикум.....	46

4.3. Варіанти завдань.....	46
4.4. Зміст звіту	50
4.5. Контрольні запитання	50
Комп'ютерний практикум № 5. ПОКАЖЧИКИ	52
5.1. Теоретичні відомості	52
Поняття покажчика	52
Операції з покажчиками	53
Особливості застосування покажчиків	56
Покажчики на покажчики	56
Типи програмної пам'яті	57
Керування динамічною пам'яттю	57
5.2. Варіанти завдань.....	61
5.3. Зміст звіту	66
5.4. Контрольні запитання.....	67
Комп'ютерний практикум № 6. ФУНКЦІЇ	68
6.1. Теоретичні відомості	68
6.2. Завдання на комп'ютерний практикум.....	72
6.3. Варіанти завдань.....	72
6.4. Зміст звіту	77
6.5. Контрольні запитання.....	77
Комп'ютерний практикум № 7. СТРУКТУРИ ТА ФАЙЛИ	78
7.1. Теоретичні відомості	78
Робота зі структурами	78
Робота з файлами	83
7.2. Завдання на комп'ютерний практикум.....	91
7.3. Варіанти завдань.....	91
7.4. Зміст звіту	94
7.5. Контрольні запитання.....	94
РЕКОМЕНДОВАНА ЛІТЕРАТУРА	95
ДОДАТОК 1. ВИМОГИ ДО ОФОРМЛЕННЯ КОДУ ПРОГРАМ НА	
МОВІ СІ	96
Угоди по ідентифікаторам.....	96
Підбір ідентифікаторів	96
Написання ідентифікаторів.....	97
Угоди по самодокументованих програмах.....	97
Коментарі.....	97
Специфікація функцій	98
Специфікація програмного файлу або модуля	99
Угоди по читабельності програм	99
"Драбинка"	99
Довжина рядків програмного тексту.....	101
Інші рекомендації.....	101

ВИМОГИ ДО ВИКОНАННЯ, ОФОРМЛЕННЯ І ЗДАЧІ КОМП'ЮТЕРНИХ ПРАКТИКУМІВ

1. Оформлення

1. Протоколи усіх комп'ютерних практикумів, необхідні ілюстративні рисунки в них, таблиці повинні бути оформлені відповідно до вимог ДСТУ 3008-2015. ЗВІТИ У СФЕРІ НАУКИ І ТЕХНІКИ. Структура і правила оформлювання (2015).
2. Рисунки схем алгоритмів у протоколі повинні зображуватися відповідно до вимог ГОСТ 19.701-90 "ЕСПД. Схеми алгоритмів, програм, даних і систем. Умовні позначення і правила виконання".
3. Перелік необхідних розділів протоколу наведено в описі кожного комп'ютерного практикуму в пункті "Зміст протоколу".
4. Приклад оформлення титульного аркуша звіту наведено на рисунку нижче:

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Кафедра Автоматизації Теплоенергетичних Процесів

Комп'ютерний практикум №1
«Умовні оператори мови Сі»
З початкової дисципліни «Програмування – 1. Процедурне програмування»
Бригада №6, варіант №26

Виконали:
Студенти групи ТА-03, ТЕФ
Богомолець А. А.
Гриценко А. О.
Помпе Д. А.

Перевірив:
Грудзинський Ю. Є.
Крилов К. В.

Київ – 2020

Рисунок 1. Титульний аркуш звіту по комп'ютерному практикуму

2. Нарахування балів

Завдання кожного комп'ютерного практикуму оцінюється в максимально можливу задану кількість балів залежно від складності завдань, згідно відповідного РСО. Кількість балів, фактично отриманих студентом за виконання комп'ютерного практикуму, залежать від ступеня відповідності виконаної програми вимогам завдання, надійності роботи програми, оформлення протоколу та власних знань студента.

Доречне прислів'я: **«що посієш, те й пожнеш»**.

Для кожного комп'ютерного практикуму встановлені терміни її виконання. В залежності від фактичних термінів здачі комп'ютерного практикуму від кількості балів, набраних студентом, віднімається штраф, встановлений РСО.

Доречне прислів'я: **«куй залізо, поки гаряче»**.

3. Самостійне виконання завдань

Комп'ютерний практикум повинен бути виконаний студентом самостійно. Випадки виявлення плагіату будуть каратися видачою нового завдання, незалежно від строку виявлення.

Доречне прислів'я: **«на чужому горбі в рай не потрапиш»**.

4. Захист завдань

Для контролю засвоєння навчальних матеріалів комп'ютерного практикуму, що здаються студентом повинні бути захищені у викладача. Нерозуміння студентом принципів роботи програми, що захищається ним, може бути розцінено, як спроба здачі чужого комп'ютерного практикуму (див. попередню нотатку).

5. Угоди про стиль кодування

Починаючи з комп'ютерного практикуму № 3 студенти повинні при написанні вихідного програмного коду неухильно дотримуватися певних правил оформлення коду на мові C, про які докладно розписано в "Стандарті кодування" GNU на сайті http://www.opennet.ru/docs/RUS/coding_standard/, і в Додатку 1 цього методичного посібника. Невідповідність тексту програми даним вимогам буде являтися причиною зниження загальної оцінки на 1...2 бали від максимальної в РСО.

6. Структура проектів

Виконаний комп'ютерний практикум повинен бути оформлений в електронному вигляді ZIP або RAR архівом і після захисту відісланий у такому вигляді викладачеві. У разі знаходження в даному архіві декількох програм, кожна з них повинна знаходитись в окремій теці.

Файли кожної програми повинні розміщуватися у двох теках: *bin* і *src* (від слова source - першоджерело).

У теці *bin* повинен розміщуватися *exe-файл* програми, яка виконується, а також допоміжні файли, необхідні для її роботи (наприклад, файли даних) і тестування. Проміжні файли, що утворюються при зборі програми, але не

потрібні для її роботи (наприклад, об'єктні файли) з програмою постачатися не повинні.

У теці *src* повинні розміщуватися вихідні файли програми, включаючи файл рішення (solution) і проектів. Середовище *Visual Studio* створює допоміжні файли з розширеннями *.opt*, *.plg*, *.ncb*, *.user* і *.suo*, при роботі проекту. Включати їх до теки *src* не потрібно.

7. Створення консольних програм в *MS Visual Studio* (на прикладі версії 2008)

Для виконання комп'ютерного практикуму розглянемо порядок створення заготовки програми в інтегрованому середовищі *Visual Studio*. Після встановлення даного середовища програмування на робочому столі (або в меню *Пуск*) з'явиться іконка для запуску з назвою «*Microsoft Visual Studio*», обираючи яку на екрані отримаємо головне вікно програми (рисунок 2).

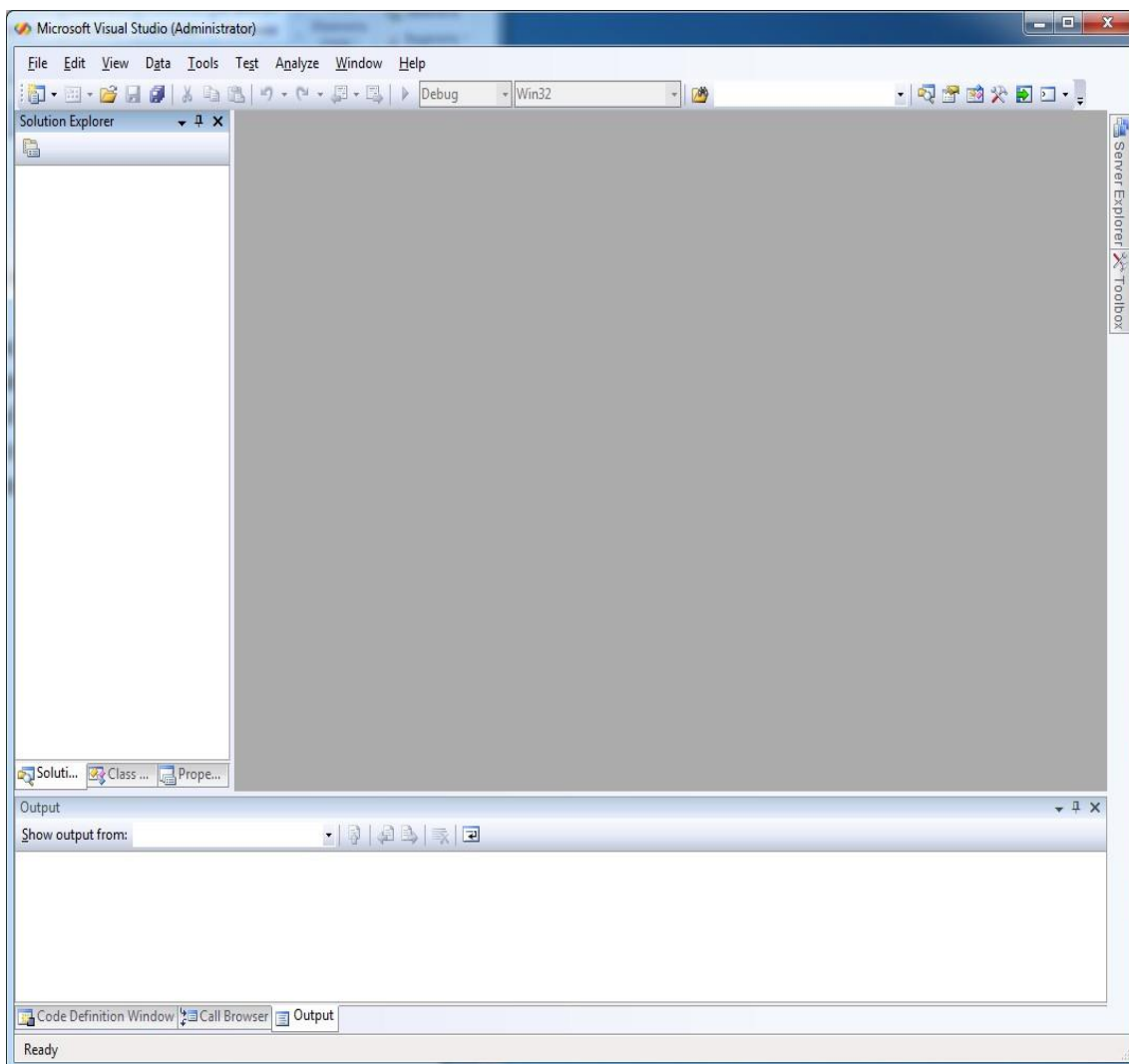


Рисунок 2. Головне вікно програми у Visual Studio 2008

Зверху вікна розташовується меню, за допомогою якого можна здійснювати управління процесом створення, компіляції та налагодження програм.

Написання програм у інтегрованому середовищі починається шляхом

створення проекту вибором меню

File -> New -> Project

після чого на екрані з'явиться діалогове вікно вибору типу проекту (Рисунок 3). В даному вікні в типах проектів (*Projects types*) слід обрати пункт *Win32*, а в шаблонах (*Templates*) *Win32 Console Application*. Всі ці пункти показані на рисунок 3.

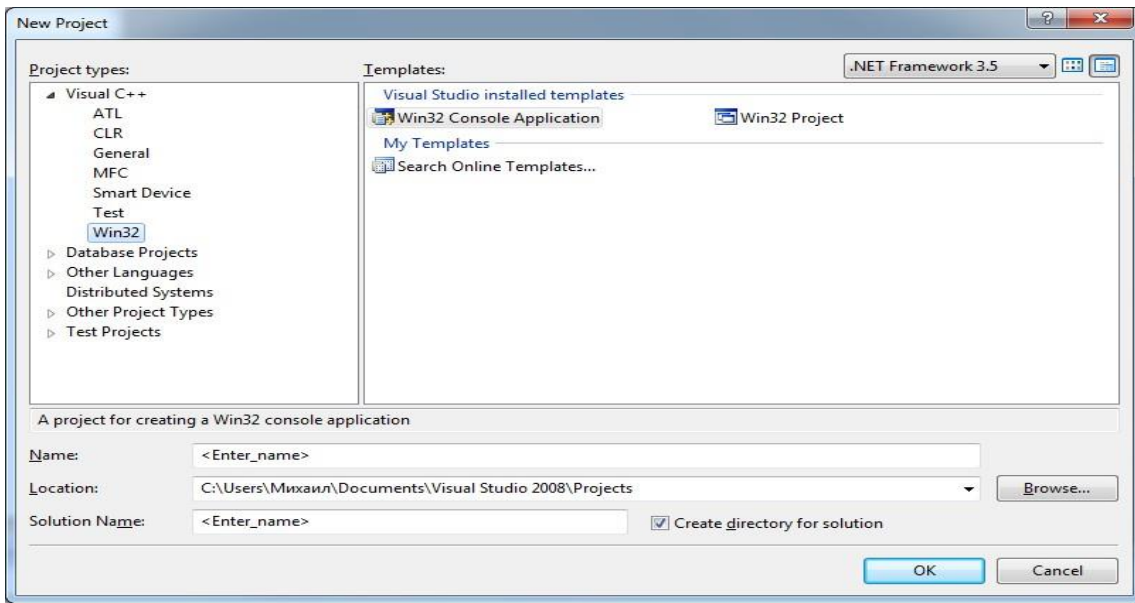


Рисунок 3. Діалогове вікно вибору типу проекту

Після цього внизу вікна в поле *Name* слід ввести ім'я проекту (англійськими літерами), наприклад, *lab1*, а в поле *Location* (розташування) вказати теку, в якій буде розташовуватися проект, наприклад, *D:\temp*. Після заповнення всіх зазначених полів і натискання на кнопку «OK» на екрані з'явиться вікно налаштування обраного консольного проекту (Рисунок 4).



Рисунок 4. Вікно налаштування консольного проекту

Тут досить натиснути на кнопку «Finish», після чого буде створено проект, а в головному вікні програми з'явиться список файлів консольного проекту (Рисунок 5)

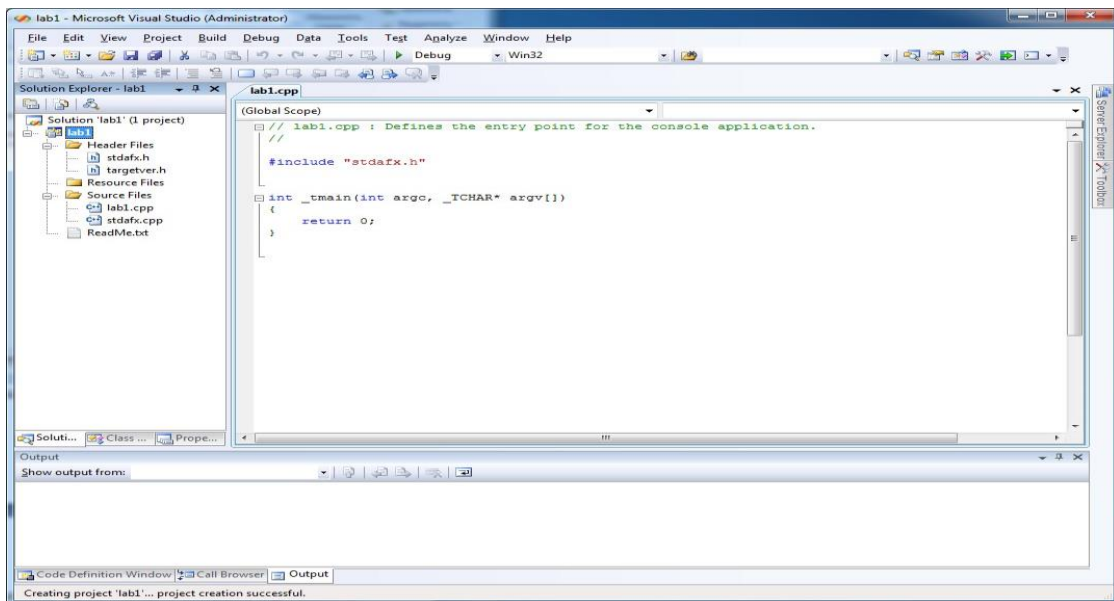


Рисунок 5. Головне вікно програми з відкритим консольним проектом

Зліва у вікні відображено список файлів проекту, з яких інтерес представляє лише файл *lab1.cpp*, тому що в ньому безпосередньо пишеться програма мовою C. Праворуч відкритий файл *lab1.cpp*, в якому використовується стандартний файл заголовка проекту *stdafx.h* і головна функція програми *_tmain()* (аналог функції *main()*). Файл *stdafx.h* слід видалити і замість нього використати всі заголовкові файли, які згадуються у пункті 0.8. Вся логіка програми повинна знаходитися всередині функції *main()*, як це описано в лекціях.

Для того щоб зберегти проект на зовнішньому носії, необхідно відкрити раніше зазначену теку при створенні проекту і в ній скопіювати каталог з назвою проекту, в даному випадку - це тека з ім'ям *lab1*.

Щоб завантажити раніше створений проект у середовище Visual Studio досить відкрити пункт меню

File-> Open-> Project / Solution

і в діалоговому вікні вказати файл проекту, що знаходиться в папці проекту, в даному випадку - це папка *lab1*.

Для запуску і компіляції програми, написаної в даному середовищі, використовується команда меню

Debug-> Start With Debugging

або комбінація клавіш **Ctrl + F5**.

Після запуску порожньої програми на екрані з'явиться вікно, показане на рисунок 6.

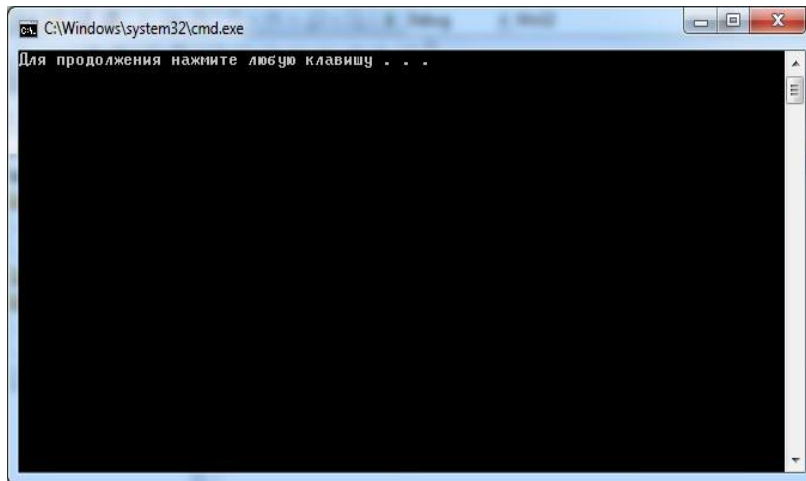


Рисунок 6. Простий консольний додаток

Таким чином можна створювати проекти для кожного комп'ютерного практикуму, зберігати, завантажувати і виконувати їх.

8. Рекомендації по створенню власних проектів в середовищі Visual Studio

Для того, щоб забезпечити безпомилкову компіляцію програми при використанні зовнішніх функцій мови C і з використанням символів кирилиці кодової таблиці 1251 в середовищі Windows студентам слід виконати наступні чотири дії для кожного свого програмного проекту:

1. Використовувати наступну структуру програмного коду програми *main()* в своїх лабораторних роботах:

```
#define _CRT_SECURE_NO_WARNINGS
#include "stdlib.h"
#include "stdio.h"
#include "conio.h"
#include "math.h"
#include "locale.h"
#include "string.h"
#include "windows.h"
...
<тип_значення_що_повертається> main (<список_параметрів>) {

<розділ_оголошень_програми>

SetConsoleCP(1251);          /* перший оператор програми, */
SetConsoleOutputCP(1251);    /* який виконується! */
```

```
...  
<виконуване_тіло_програми_(код)>
```

```
...  
}
```

2. Далі слід вказати курсором миші на ім'я проекту у вікні *Solution Explorer*, клацнути правою клавішею миші і вибрати у випадаючому меню пункт *"Properties"*. У вікні властивостей, ліворуч, зробити наступний вибір:

"Configuration Properties" -> "General"

і на вкладці праворуч перейти до властивості *"Character Set"*. В цій властивості, за допомогою меню, що випадає встановити набір символів **"Not Set"**.

3. Далі слід, вказавши курсором мишки на ім'я проекту у вікні *Solution Explorer*, клацнути правою клавішею миші та обрати у випадаючому меню пункт *"Properties"*. У вікні властивостей зліва, зробити наступний вибір:

"Configuration Properties" -> "C/C++" -> "Pre-compiled headers"

і на вкладці праворуч перейти до властивості *"Pre-compiled header"*. У цій властивості, за допомогою меню, що випадає встановити режим *"Do not use precompiled headers"*.

4. У властивостях нового проекту для даних лабораторних робіт слід вказати явно використання стандарту мови C++ при компіляції, що дозволить повноцінно використовувати нові можливості C++ при написанні коду C. Для цього слід вказати курсором мишки на ім'я проекту у вікні *Solution Explorer*, клацнути правою клавішею миші і вибрати в меню, що випало, пункт *"Properties"*. У вікні властивостей, ліворуч, зробити наступний вибір:

Configuration Properties -> C/C++ -> Advanced

У правій частині вікна, що відкрилося, в рядку вибору *"Compile as"* вибрати режим *"Compile as C++ code (/ TC)"* і натиснути внизу кнопку ОК. Виконати цю процедуру для кожного проекту в рішенні.

У деяких комп'ютерних практикумах потрібно вводити за допомогою функції *scanf()* рядки символів, всередині яких можуть бути присутні пробіли та інші розділювачі слів (табуляція та т.ін.) Для того, щоб уникнути введення тільки першого слова рядка (до розділювача) студенту слід використовувати наступний вид функції *scanf()*:

```
scanf ("%[^\\n]s", str);
```

У багатьох комп'ютерних практикумах потрібно виконати кілька завдань, тобто написати кілька непов'язаних між собою програм. У цьому випадку студенту необхідно створити в межах одного рішення Visual Studio кілька

проектів (див. [7]), а не створювати кілька рішень. У цьому випадку він отримує можливість простого переходу між проектами, не закриваючи створеного рішення і не "плодячи" рішень, тим самим, спрощуючи собі роботу і не "захаращуючи" жорсткий диск даними.

При виконанні кожного комп'ютерного практикуму (крім першого та другого) студент повинен передбачати у своєму програмному коді відповідні перевірки на допустимість вхідних даних реальному діапазону значень.

Кожна з написаних студентом програм (крім першої) повинна бути інтерактивною і на кожному етапі свого виконання активно взаємодіяти з користувачем, інформуючи його про необхідні вхідні дані, результати свого виконання і варіантах подальшої поведінки користувача.

Програмний код написаних студентом лабораторних робіт не повинен містити фрагменти, що не стосуються теми комп'ютерного практикуму, варіанту завдання комп'ютерного практикуму або невживані фрагменти, (окрім тексту коментарів в програмі).

Комп'ютерний практикум № 1. УМОВНІ ОПЕРАТОРИ МОВИ C

Мета комп'ютерного практикуму: вивчити особливості використання умовних операторів *if* і *switch*.

1.1. Теоретичні відомості

Умовний оператор *if*

Для того, щоб мати можливість реалізувати логіку, в програмі використовуються умовні оператори. Ці оператори можна представити у вигляді вузлових пунктів, досягаючи яких програма робить вибір, за яким з можливих напрямків рухатися далі. Наприклад, потрібно визначити, чи містить деяка змінна *arg* позитивне чи негативне число і вивести відповідне повідомлення на екран. Для цього можна скористатися оператором *if* (якщо), який і виконує подібні перевірки.

У самому простому випадку синтаксис цього оператора *if* наступний:

if (вираз) <оператор>

Якщо значення параметра «вираз» дорівнює «істинно», виконується оператор, інакше він пропускається програмою. Слід зазначити, що «вираз» є умовним виразом, в якому виконується перевірка деякої умови. У табл. 1.1 представлені варіанти простих логічних виразів оператора *if*.

Таблиця 1.1. Прості логічні вирази

<i>if</i> ($a < b$)	Істинно, якщо змінна <i>a</i> менше змінної <i>b</i> і хибне в іншому випадку.
<i>if</i> ($a > b$)	Істинно, якщо змінна <i>a</i> більше змінної <i>b</i> і хибне в іншому випадку.
<i>if</i> ($a == b$)	Істинно, якщо змінна <i>a</i> дорівнює змінній <i>b</i> і хибне в іншому випадку.
<i>if</i> ($a <= b$)	Істинно, якщо змінна <i>a</i> менше або дорівнює змінній <i>b</i> і хибне в іншому випадку.
<i>if</i> ($a >= b$)	Істинно, якщо змінна <i>a</i> більше або дорівнює змінній <i>b</i> і хибне в іншому випадку.
<i>if</i> ($a != b$)	Істинно, якщо змінна <i>a</i> не дорівнює змінній <i>b</i> і хибне в іншому випадку.
<i>if</i> (<i>a</i>)	Істинно, якщо змінна <i>a</i> не дорівнює нулю, і хибне в іншому випадку.

Наведемо приклад використання умовного оператора *if*. Наступна програма

дозволяє визначати знак введеної змінної.

Лістинг 1.1. Програма визначення знаку введеного числа.

```
#include <stdio.h>

int main () {
    float x;
    printf ("Введіть число:");
    scanf ("%f", &x);
    if (x < 0)
        printf ("Введене число %f є негативним. \n", x);
    if (x >= 0)
        printf ("Введене число %f є додатнім. \n", x);
    return 0;
}
```

Аналіз наведеного тексту програми показує, що два умовних оператора можна замінити одним, використовуючи конструкцію

```
if (вираз)
    <оператор1>
else
    <оператор2>
```

яка інтерпретується таким чином. Якщо «вираз» істинний, то виконується «оператор1», інакше виконується «оператор2».

У випадках, коли при виконанні якої-небудь умови необхідно записати більше одного оператора, необхідно використовувати оператор блоку (фігурні дужки), тобто використовувати конструкцію виду

```
if (вираз) {
    <список операторів>
}
else {
    <список операторів>
}
```

Слід зазначити, що після ключового слова *else* формально можна поставити ще один оператор умови *if*, в результаті отримаємо ще більш гнучку конструкцію умовних переходів:

```
if (вираз1)
    <оператор1>
else if (вираз2)
    <оператор2>
else
```

<оператор3>

До цих пір розглядалися прості умови типу ($x < 0$). Разом з тим оператор *if* дозволяє реалізовувати більш складні умовні переходи. У мові C є три логічні операції:

&& - Логічне І

|| - Логічне АБО

! - Логічне НІ

На основі цих трьох логічних операцій можна сформуванати більш складні умови. Наприклад, якщо є три змінні *exp1*, *exp2* і *exp3*, то вони можуть становити логічні конструкції, представлені в табл. 1.2.

Таблиця 1.2. Приклад складних логічних виразів

<i>if (exp1 > exp2 && exp2 < exp3)</i>	Істинно, якщо значення змінної <i>exp1</i> більше значення змінної <i>exp2</i> і значення змінної <i>exp2</i> менше значення змінної <i>exp3</i> .
<i>if (exp1 <= exp2 exp1 >= exp3)</i>	Істинно, якщо значення змінної <i>exp1</i> менше або дорівнює значенню змінної <i>exp2</i> або значення змінної <i>exp2</i> більше або дорівнює значенню змінної <i>exp3</i> .
<i>if (exp1 && exp2 && !exp3)</i>	Істинно, якщо одночасно істинні значення <i>exp1</i> , значення <i>exp2</i> і хибне значення <i>exp3</i> .
<i>if (!exp1 !exp2 && exp3)</i>	Істинно, якщо хибне значення <i>exp1</i> або одночасно хибне значення <i>exp2</i> і істинне значення <i>exp3</i> .

Подібно операціями множення і додавання в математиці, логічні операції І АБО НІ, також мають свої пріоритети. Найвищий пріоритет має операція НІ, тобто така операція виконується у першу чергу. Нижчий пріоритет у операції І, і нарешті найменший пріоритет в операції АБО.

Оператор-перемикач *switch*

Умовна операція *if* полегшує написання програм, в яких необхідно проводити вибір між невеликим числом можливих варіантів. Однак іноді в програмі необхідно здійснити вибір одного варіанта з безлічі можливих. Формально для цього можна скористатися вкладеною конструкцією *if else if ... else*. Однак у багатьох випадках виявляється більш зручним застосовувати оператор-перемикач *switch* мови C. Синтаксис цього оператора наступний:

```
switch (змінна) {  
    case константа1:  
        <група_операторів 1>  
    case константа2:  
        <група_операторів 2>
```

```

...
default:
    <група_операторів default>

}

```

Даний оператор послідовно перевіряє на рівність *змінної* константам, що стоять після ключового слова *case*. Якщо жодна з констант не дорівнює значенню змінної, то виконуються оператори, що знаходяться після слова *default*. Оператор *switch* має таку особливість. Припустимо, значення змінної дорівнює значенню *константа1* і виконуються оператори, що стоять після першого ключового слова *case*. Після цього, якщо не вжити заходів, виконання програми "провалиться" нижче, на блок з *константа2*, і т.ін., по всіх блоках *case*. Щоб уникнути такої ситуації після закінчення блоку операторів в *case* слід використовувати останнім оператором блоку оператор *break* для переходу програми до наступного оператору після *switch*.

У лістингу 1.2 наведено приклад програмування оператора-перемикача *switch*.

Лістинг 1.2. Приклад використання оператора *switch*.

```

#include <stdio.h>
int main () {
    int x;
    char ch;
    printf ("Введіть число:");
    scanf ("%d", &x);
    switch (x) {
        case 1:
            printf ("Введено число 1 \n");
            break;
        case 2:
            printf ("Введено число 2 \n");
            break;
        default:
            printf ("Введено інше число \n");
    }
    printf ("Введіть символ:");
    scanf ("%c", &ch);
    switch (ch) {
        case 'a':
            printf ("Введено символ a \n");
            break;
        case 'b':
            printf ("Введено символ b \n");
            break;
        default:
            printf ("Введено інший символ \n");
    }
    return 0;
}

```

Даний приклад демонструє два різних варіанти використання оператора *switch*. У першому випадку виконується аналіз введеної цифри, у другому - аналіз введеного символу. Слід зазначити, що даний оператор може робити вибір тільки на підставі рівності свого аргументу одному з перерахованих значень *case*, тобто перевірка виразів типу $(x < 0)$ в даному випадку неможлива.

1.2. Завдання на комп'ютерний практикум

1. Написати програму роботи з умовним оператором *if* у відповідності з номером свого варіанта.
2. Написати програму з використанням оператора *switch* у відповідності з номером свого варіанта.
3. Зробити висновки про отримані результати роботи програм.

1.3. Варіанти завдань

Варіант	Оператор <i>if</i>	Оператор <i>switch</i>
1	Написати програму обчислення модуля введеного числа	Написати програму переведення введеного символу від 'a' до 'f' у верхній регістр
2	Написати програму перевірки входження введеного числа в діапазон від -2 до 2	Написати програму переведення введеного символу від 'A' до 'F' у нижній регістр
3	Написати програму перевірки не входження введеного числа в діапазон від 0 до 5	Написати програму заміни введеної цифри від 0 до 9 відповідним словом
4	Написати програму перевірки на додатність введеного числа	Написати програму, яка по номеру квартири видає прізвище її мешканця. Розглянути в програмі не менше 6 квартир
5	Написати програму перевірки на від'ємність введеного числа	Написати програму, яка по номеру місяця видає кількість днів в місяці
6	Написати програму визначення знаку введеного числа	Написати програму, яка по порядковому номеру місяця видає його назву

Варіант	Оператор <i>if</i>	Оператор <i>switch</i>
7	Написати програму перевірки входження введеного числа в діапазон від -6 до -2	Написати програму, яка по порядковому номеру місяця видає пору року, до якої він належить (зима, весна, літо, осінь)
8	Написати програму перевірки не входження введеного числа в діапазон від -5 до -1	Скласти програму, яка б видавала назву місяця, наступного за введеним номером місяця (враховувати, що за груднем йде січень)
9	Написати програму обчислення суми модулів двох введених чисел	Скласти програму, яка б по введеному числу K (від 1 до 10) видавала б римську цифру цього числа
10	Написати програму обчислення $\frac{1}{a}$ з перевіркою умови $a \neq 0$	Написати програму порівняння введеного символу з 'a', 's', 'd', 'j' і 'e'
11	Написати програму обчислення $\sqrt{a}$ з перевіркою умови $a \geq 0$	Для цілого числа K від 1 до 10 надрукувати повністю словами фразу "мені K років", враховуючи, що при деяких K для слова "років" слід враховувати відмінкові закінчення
12	Написати програму перевірки потрапляння введеного числа в діапазони від -3 до 0, від 3 до 6	Для цілого числа K від 1 до 10 надрукувати повністю словами фразу "ми знайшли K грибів у лісі", погодивши закінчення слова гриб з числом K
13	Написати програму перевірки потрапляння введеного числа в діапазони від 4 до 7, від 1 до 11	Написати програму визначення дня тижня за його номером
14	Написати програму перевірки введеного числа з плаваючою точкою: ціле воно, чи дробове.	Написати програму обчислення результату математичної операції (+, -, * чи /) над двома числами в залежності від введеного символу, що відповідає необхідній операції

Варіант	Оператор <i>if</i>	Оператор <i>switch</i>
15	Написати програму перевірки ділення без остачі двох введених чисел	Написати програму перевірки, чи є введене число простим для діапазону від 0 до 100
16	З введених двох чисел з різною парністю вивести на екран непарне	Написати програму перекладу 12-бальної шкали оцінювання у 5-бальну
17	Написати програму перевірки співпадіння будь-яких 2 із 3 введених чисел	Мастям гральних карт умовно присвоєні такі порядкові номери: масті «піки» - 1, масті «трефи» - 2, масті «бубни» - 3, масті «черви» - 4, а номери карт: «валету» - 11, «дамі» - 12, «королю» - 13, «тузу» - 14 (порядкові номери інших карт відповідають їх назвам: «шістка», «дев'ятка», тощо). За заданим номером масті m ($1 \leq m \leq 4$) і номеру карти k ($6 \leq k \leq 14$) визначити повну назву (масть і номер) відповідної карти у вигляді «Дама пік», «Шістка бубон», тощо
18	Написати програму визначення середнього числа із трьох введених	Написати програму перетворення номера місяця у його назву в родовому відмінку
19	Написати програму перевірки належності точки колу з центром в точці (x,y) і радіусом R	Написати програму визначення кількості днів у місяці за його номером
20	Написати програму перевірки існування трикутника за введеними довжинами трьох його сторін	Написати програму перетворення слова студент в коректне кількісне представлення залежно від введеного числа від 1 до 9 (1 студент, 2 студенти, 5 студентів)

Варіант	Оператор <i>if</i>	Оператор <i>switch</i>
21	Ввести з клавіатури координати точок А (x_0, y_0) і В (x_1, y_1). Визначити, яка з точок А чи В найбільш віддалена від початку координат	Написати програму визначення ноти за її символом (англійська маленька літера)
22	Ввести з клавіатури два числа. Визначити, що більше, сума квадратів або квадрат суми цих чисел	Написати програму визначення кількості днів у місяці за його номером. Розглянути два випадки: 1. рік не є високосним 2. рік може бути високосним
23	Ввести з клавіатури значення трьох сторін трикутника а, b і с і визначити, чи є він прямокутним	Мастям гральних карт умовно присвоєні такі порядкові номери: "піка" - 1, "трефа" - 2, "бубна" - 3, "чирва" - 4. По заданому номеру масті визначити її назву
24	Ввести з клавіатури три числа, додатні звести в квадрат, а з від'ємних взяти кубічний корінь	Поштовий автомат пропонує вітальні листівки на три теми в трьох варіантах. Ввести з клавіатури номер теми (від 1 до 3) і номер варіанта (А, В, С), видати потрібну листівку (Повідомлення, наприклад: «Новорічні, варіант А»)
25	Ввести з клавіатури координати точки А (x, y). Визначити, в якій чверті координатної площини лежить дана точка	Написати програму аналізу введеної великої та маленької літери кирилиці (голосна, приголосна)
26	Ввести з клавіатури три числа, додатні звести в куб, а від'ємні замінити на 0	Написати програму визначення декади (перша, друга, третя) місяця в яку потрапляє введене число

Варіант	Оператор <i>if</i>	Оператор <i>switch</i>
27	Відомі дві швидкості, одна в кілометрах в годину, інша в метрах в секунду. Яка з них більше?	Написати програму, яка за кодом міста і тривалості переговорів обчислює їх вартість і результат виводить на екран: Одеса-код 048, 15грн; Київ-код 044, 18грн; Харків-код 057, 13грн; Вінниця-код 045, 11грн.
28	Визначити, чи є точка з координатами A (X, Y) точкою перетину діагоналей квадрата зі стороною R, одна з вершин якого розташована на початку координат	Написати програму, яка в залежності від характеру вітру видає повідомлення про його швидкість: від 1 до 4 м / с - слабкий (1); від 5-10 м / с - помірний (2); від 9-18 м / с - сильний (3); більше 19 м / с - ураганний (4)
29	Відомі значення опору і напруги для двох не поєднаних одна з одною ділянок електричного ланцюга. Визначити, на якій ділянці протікає менший струм?	За введеною датою народження (два числа -день, місяць) визначити знак Зодіака людини
30	Ввести з клавіатури поточний час і визначити час доби (pm - з 0 до 12, am - з 12 до 24)	За введеною кількістю отриманих балів за предмет студентом за семестр визначити його оцінку у вигляді: <i>відмінно, дуже добре, добре, задовільно, достатньо, незадовільно</i>

1.4. Зміст звіту

1. Титульний лист з назвою комп'ютерного практикуму, номером варіанту, прізвищем студента та номером групи.
2. Мета комп'ютерного практикуму.
3. Короткі теоретичні відомості.
4. Завдання.
5. Схеми алгоритмів.
6. Тексти програм.
7. Результати роботи програм.
8. Висновки про отримані результати роботи програм.

1.5. Контрольні питання

1. Запишіть умовний оператор *if* для визначення знаку змінної *var*.
2. У яких випадках слід використовувати оператор *switch*?
3. Використовуючи умовний оператор, виконайте перевірку на належність значення змінній діапазону [10; 20).
4. Наведіть програму заміни малих латинських букв великими з використанням оператора *switch*.
5. Як записується логічна рівність в операторі *if*?
6. Наведіть позначення логічного знаку «не дорівнює».
7. Якими символами позначаються логічні операції І і АБО в умовному операторі *if*?

Комп'ютерний практикум № 2. ОПЕРАТОРИ ЦИКЛІВ МОВИ C

Мета комп'ютерного практикуму: вивчити особливості використання операторів циклу *while*, *for* і *do while*.

2.1. Теоретичні відомості

Часто при створенні програм на *ЕОМ* потрібно багато разів виконати одну і ту саму групу операторів декілька разів. Наприклад, для обчислення суми ряду довжиною *N* або перебору елементів масиву з метою визначення найбільшого або найменшого значення і т.ін. У всіх цих випадках необхідний інструмент для реалізації повторюваних операцій і таким інструментом є оператори циклу.

Оператор циклу *while*

За допомогою даного оператора реалізується цикл, який виконується до тих пір, поки істинна умова циклу. Синтаксис цього оператора наступний:

```
while (<умова>) {  
    <тіло циклу>  
}
```

Наведемо приклад реалізації даного циклу, в якому виконується підсумовування елементів ряду

$$S = \sum_{i=0}^{\infty} i$$

поки $S < N$. Фрагмент програмного коду наведено нижче:

```
int N=20,  
    i = 0;  
int S = 0;  
while (S < N) {  
    S=S+i;  
    i++;  
}
```

У даному прикладі реалізується цикл *while* з умовою $i < N$. Оскільки початкове значення змінної $i = 0$, а $N = 20$, то умова істинна і виконується тіло циклу, в якому здійснюється додавання змінної i та збільшення її значення на 1. Очевидно, що після 7 ітерації значення $S = 15$, а значення $i = 6$, умова стане хибною і цикл буде завершений. Продемонструємо гнучкість мови C, змінивши даний приклад наступним чином:

```
int N = 20, i = 0;  
int S = 0;  
while ((S = S + i++) < N);
```

В даному випадку при перевірці умови спочатку виконуються оператори, що стоять в дужках, де і здійснюється підсумовування елементів ряду і тільки, потім, перевіряється умова. Результат виконання обох варіантів програм однаковий і $S = 21$. Проте остання конструкція буває зручною при реалізації

опитування клавіатури, наприклад, за допомогою функції `scanf()`:

```
int num;
while (scanf ("%d", &num) == 1)
    printf ("Ви ввели значення %d \n", num);
```

Даний цикл буде працювати, поки користувач вводить цілочисельні значення і зупиняється, якщо введена буква або дійсне число. Слід зазначити, що цикл `while` можна примусово завершити навіть при істинній умови циклу. Це досягається шляхом використання оператора `break`. Перепишемо попередній приклад так, щоб цикл завершувався, якщо користувач введе число 0.

```
int num;
while (scanf ("%d", &num) == 1) {
    if (!num)
        break;
    printf ("Ви ввели значення %d \n", num);
}
```

Цикл завершується відразу після використання оператора `break`, тобто у наведеному прикладі, при введенні з клавіатури нуля функція `printf()` виконуватися не буде і програма перейде на наступний оператор після `while`. Таким чином видно, що в якості умови можливі такі ж конструкції, що і в окремому операторі.

Оператор циклу `for`

Робота оператора циклу `for` подібна оператору `while` з тією лише різницею, що оператор `for` передбачає зміну значення деякої змінної, що називається лічильником циклу і перевірки умови виконання циклу на істинність. Робота даного оператора продовжується до тих пір, поки істинна умова циклу. Синтаксис оператора `for` наступний:

```
for (<ініціалізація лічильника>; <умова>; <зміна значення лічильника>) {
    <тіло циклу>
}
```

Розглянемо особливість реалізації даного оператора на прикладі виведення таблиці кодів ASCII нижнього регістру символів латиниці:

```
char ch;
for (ch = 'a'; ch <= 'z'; ch ++){
    printf ("Значення ASCII для %c -%d. \n", ch, ch);
```

У даному прикладі в якості лічильника циклу виступає змінна `ch`, яка ініціалізується символом 'a'. Це означає, що в змінну `ch` заноситься число 97 - код символу 'a'. Саме так символи представляються в пам'яті комп'ютера. Код символу 'z' - 122, і всі малі букви латинського алфавіту мають коди в діапазоні [97; 122]. Тому, збільшуючи значення `ch` на одиницю, отримуємо код наступної букви, яка виводиться за допомогою функції `printf()`. Враховуючи все

вищесказане, цей же приклад можна було б записати наступним чином:

```
for ( ch = 97; ch <= 122; ch ++)  
    printf ("Значення ASCII для %c -%d. \n", ch, ch);
```

Існує багато особливостей реалізації даного оператора, відзначимо основні з них, які можуть помітно підвищити швидкість написання програм. Наступним прикладом продемонструємо особливості зміни значення лічильника циклу.

```
int line_cnt = 1;  
double debet;  
for (debet = 100.0; debet < 150.0; debet = debet* 1.1, line_cnt ++)  
    printf ("%d. Ваш борг тепер дорівнює %.2f. \n", line_cnt, debet);
```

Наступний фрагмент програми демонструє можливість програмування складної умови всередині циклу.

```
int exit = 1;  
for (num = 0; num < 100 && !exit; num += 1) {  
    scanf ("%d", &mov);  
    if (!mov)  
        exit = 0;  
    printf ("Добуток num*mov = %d. \n", num*mov);  
}
```

Кожна з секцій оператора *for* чи усі разом можуть бути відсутніми.

Оператор *for*, в якому з усіх секцій залишилася тільки секція умови:

```
int i = 0;  
for (;i < 100;)  
    i++;
```

Оператор *for*, при повній відсутності усіх секцій:

```
int i = 0;  
for (;;;){  
    i ++;  
    if (i> 100)  
        break;  
}
```

В останньому прикладі оператор *break* служить для виходу з циклу *for*, тому що інакше цикл працюватиме «вічно» (не маючи коректних умов виходу).

Оператор циклу *do while*

Всі представлені вище оператори циклів, так чи інакше, перевіряють умову перед виконанням циклу, завдяки чому існує ймовірність, що оператори

всередині циклу ніколи не будуть виконані (якщо умова циклу не виконається ще при першому заході до циклу). Такі цикли називають циклами з передумовою. Проте бувають ситуації, коли доцільно виконувати перевірку умови після того, як будуть виконані оператори, які стоять всередині циклу. Це досягається шляхом використання операторів циклу *do while*, які реалізують цикл з постумовою. Наступний приклад демонструє реалізацію такого циклу.

```
const int secret_code = 13;
int code_ent;
do {
    printf ("Введіть секретний код:");
    scanf ("%d", &code_ent);
}
while (code_ent != secret_code);
```

З наведеного прикладу видно, що цикл з постумовою працює поки істинна умова, тобто в даному випадку, поки значення введеного коду буде відрізнятися від значення секретного коду. Також слід звернути увагу на те, що в кінці оператора *do while* (після умови циклу) повинна стояти крапка з комою. При реалізації даного циклу можна використовувати складні умови, подібно циклу *while*, а також примусово виходити з циклу за допомогою оператора *break*.

Програмування вкладених циклів

Всі розглянуті вище оператори циклів допускають використання будь-яких інших операторів мови C всередині циклу, в тому числі і операторів циклу. Це означає, що всередині одного циклу може знаходитись інший, що призводить до реалізації вкладених циклів. Вкладені цикли необхідні для вирішення великої кількості завдань, наприклад, обчислення подвійних, потрійних та інших видів сум, перегляд елементів двовимірного масиву і багатьох інших завдань. Як приклад вкладених циклів розглянемо задачу обчислення суми подвійного ряду

$$S = \sum_{i=0}^N \sum_{j=0}^M i * j$$

```
int S = 0;
int M = 10, N = 5;
for (i = 0; i <= N; i++)
    for (j = 0; j <= M; j++)
        S += i*j;
```

Того ж результату можна досягти за допомогою циклу *while*.

2.2. Завдання на комп'ютерний практикум

1. Написати програму роботи з операторами циклів *while* та *for* відповідно до номера свого варіанта.

2. Написати програму з використанням оператора циклу *do while* у відповідності з номером свого варіанта.
3. Зробити висновки про отримані результати роботи програм.

2.3. Варіанти завдань

Варіант	Оператори циклів <i>while</i> і <i>for</i>	Оператор цикла <i>do while</i>
1	Обчислити $\sum_{i=1}^N \frac{1}{i^2}$ з використанням циклу <i>for</i> . Число N вводиться з клавіатури	Написати програму вводу довільних чисел до тих пір, поки не буде введено число 0
2	Обчислити $f(x) = kx+b$ при $x=\overline{1, N}$ з використанням циклу <i>while</i> . Числа N , k і b вводяться з клавіатури	Написати програму введення довільних символів до тих пір, поки не буде введений символ q
3	Обчислити $\sum_{i=1}^N \sum_{j=1}^M (i + j)$ за допомогою вкладених циклів <i>for</i> . Числа N і M вводяться з клавіатури	Написати програму підрахунку суми з 10 чисел, що вводяться з клавіатури
4	Обчислити суму $S = \sum_{i=1}^{\infty} i$, поки $S < N$ за допомогою циклу <i>while</i> . N вводиться з клавіатури	Написати програму обчислення добутку 5 чисел, введених з клавіатури
5	Обчислити суму $S = \sum_{i=1}^{\infty} i^2$, поки $S < N$ за допомогою циклу <i>for</i>	Написати програму обчислення модулів введених чисел до тих пір, поки користувач не введе 0
6	Обчислити $\sum_{i=1}^N \sum_{j=1}^M (1/(i + j))$, за допомогою вкладених циклів <i>while</i> . Числа N і M вводяться з клавіатури	Написати програму визначення знака введених чисел до тих пір, поки користувач не введе 0
7	Обчислити $f(x) = x^2 + b$, при $x = \overline{-N, N}$ з використанням циклу <i>for</i> . N вводиться з клавіатури	Написати програму визначення мінімального введеного числа з N чисел
8	Обчислити $\sum_{i=-10}^N (1/i^3)$, $i \neq 0$ з використанням циклу <i>for</i> . N вводиться з клавіатури	Написати програму визначення максимального введеного числа з N чисел

Варіант	Оператори циклів <i>while</i> і <i>for</i>	Оператор цикла <i>do while</i>
9	Обчислити $\sum_{i=-N}^M \sum_{j=0}^N (1/(i+j)^2)$, $(i+j) \neq 0$, з допомогою вкладених циклів <i>for</i> . N та M вводяться з клавіатури	Написати програму визначення мінімального серед додатніх введених M чисел
10	Обчислити $f(x) = 1/x$, $x \neq 0$, $x = -N, N$ з використанням циклу <i>for</i> . N вводиться з клавіатури	Написати програму визначення максимального серед від'ємних введених M чисел
11	Обчислити $\sum_{i=0}^n (2i+1)$, За допомогою циклу <i>for</i> . n вводиться з клавіатури	Написати програму, яка "загадує" число від 1 до M і пропонує користувачу вгадати його за 5 спроб
12	Написати програму, яка обчислює факторіал введеного з клавіатури числа. За допомогою циклу <i>for</i>	Написати програму визначення мінімального введеного числа з N чисел
13	Написати програму, яка виводить на екран фрагмент таблиці множення введеного з клавіатури числа. За допомогою циклу <i>while</i>	Написати програму вводу числа π , поки воно не буде правильно введено з точністю до ε . Число ε задати з клавіатури
14	Написати програму, яка виводить на екран зображення шахматної дошки. Чорні клітинки відображати *, білі – пробілом. За допомогою циклу <i>while</i>	Написати програму вводу чисел, яка буде додавати введені числа до тих пір, поки їх сума не стане більше за M .
15	Написати програму, яка виводить на екран таблицю значень функції $y = -2.4x^2 + 5x - 3$ у діапазоні від $-N$ до N з кроком h . За допомогою циклу <i>for</i> . N та h задати з клавіатури	Написати програму вводу числа e , поки воно не буде правильно введено з точністю до ε . Число ε задати з клавіатури

Варіант	Оператори циклів <i>while</i> і <i>for</i>	Оператор цикла <i>do while</i>
16	Написати програму, яка виводить на екран таблицю значень функції $y= x $ в діапазоні від $-N$ до N з кроком h . За допомогою циклу <i>for</i> . N та h задати з клавіатури	Написати програму вводу довільних чисел до тих пір, поки не буде введено число більше за M
17	Написати програму, яка виводить на екран таблицю значень функції $y= x-2 + x+1 $ у діапазоні від $-N$ до N з кроком h . За допомогою циклу <i>for</i> . N та h задати з клавіатури	Написати програму визначення суми модулів введених M чисел
18	Написати програму, яка обчислює середнє арифметичне послідовності дробових чисел, що вводяться з клавіатури. Кількість чисел повинна задаватися під час роботи програми. За допомогою циклу <i>for</i>	Написати програму, яка "загадує" букву від C_1 до C_2 і пропонує користувачу вгадати літеру за 5 спроб. Літери C_1 та C_2 вводяться з клавіатури
19	Написати програму, яка обчислює середнє арифметичне послідовності дробових чисел, що вводяться з клавіатури. Кількість чисел повинна задаватися під час роботи програми. За допомогою циклу <i>while</i>	Написати програму, яка виводить на екран таблицю значень функції $y = -2.4 \cdot x^2 + 5 \cdot x - 3$ у діапазоні від $-N$ до N з кроком h . N та h задати з клавіатури
20	Написати програму, яка виводить на екран таблицю степенів двійки від нульової до N . За допомогою циклу <i>while</i> . N вводиться з клавіатури	Написати програму обчислення добутку перших M від'ємних введених чисел
21	Одна штука деякого товару коштує V грн. Надрукувати таблицю вартості 2, 3, ..., N штук цього товару. N та V вводяться з клавіатури	Знайти з точністю N знаків після коми значення виразу: $\sum_{i=1}^{\infty} \frac{1}{i^2}$

Варіант	Оператори циклів <i>while</i> і <i>for</i>	Оператор цикла <i>do while</i>
22	Надрукувати таблицю відповідності між вагою в фунтах і вагою в кілограмах для значень від $Fmin$ до $Fmax$ фунтів з кроком H . $Fmin$, $Fmax$ та H вводяться з клавіатури	Знайти найбільше додатне n , для якого виконується умова $3n^2 - 730n < 0$
23	Надрукувати таблицю переведення відстаней в дюймах в сантиметри для значень від $Lmin$ до $Lmax$ дюймів з кроком H (1 дюйм = 25,4 мм). $Lmin$, $Lmax$ та H вводяться з клавіатури	За допомогою розкладу функції, наведеної нижче у ряд Тейлора обчислити з точністю $\varepsilon > 0$ її значення для заданого значення x $y = \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$ Значення ε та x вводяться з клавіатури
24	Надрукувати таблицю переведення від $Dmin$ до $Dmax$ доларів США з кроком H за поточним курсом K . Значення $Dmin$, $Dmax$, H та K вводяться з клавіатури.	За допомогою розкладу функції, наведеної нижче у ряд Тейлора обчислити з точністю $\varepsilon > 0$ її значення для заданого значення x $y = \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \dots, (x < 1);$ Значення ε та x вводяться з клавіатури
25	Надрукувати "стовпчиком" значення $\sin(Lmin)$, ..., $\sin(Lmax)$ з кроком H . У дужках – градуси. $Lmin$, $Lmax$ та H вводяться з клавіатури	За допомогою розкладу функції, наведеної нижче у ряд Тейлора обчислити з точністю $\varepsilon > 0$ її значення для заданого значення x $y = \frac{1}{1+x} = 1 - x + x^2 - x^3 + \dots, (x < 1);$ Значення ε та x вводяться з клавіатури
26	Дано натуральне число N . Визначити: а) Скільки цифр в числі N . б) Чому дорівнює сума його цифр? в) Знайти першу цифру числа N .	За допомогою розкладу функції, наведеної нижче у ряд Тейлора обчислити з точністю $\varepsilon > 0$ її значення для заданого значення x $y = \ln \frac{1+x}{1-x} = 2 \cdot \left(\frac{x}{1} + \frac{x^3}{3} + \frac{x^5}{5} + \dots \right), (x < 1);$ Значення ε та x вводяться з клавіатури

Варіант	Оператори циклів <i>while</i> і <i>for</i>	Оператор цикла <i>do while</i>
27	Знайти знакозмінну суму цифр запису числа N (нехай запис N в десятковій системі є $\alpha_k \alpha_{k-1} \dots \alpha_0$; знайти $\alpha_k - \alpha_{k-1} + \alpha_{k-2} + \dots + (-1)^k \alpha_0$)	За допомогою розкладу функції, наведеної нижче у ряд Тейлора обчислити з точністю $\varepsilon > 0$ її значення для заданого значення x $y = \sqrt{1+x} = 1 + \frac{1}{2} \cdot x - \frac{1}{2 \cdot 4} \cdot x^2 + \frac{1 \cdot 3}{2 \cdot 4 \cdot 6} \cdot x^3 - \dots, (x < 1)$ Значення ε та x вводяться з клавіатури
28	Дано натуральне число N . Отримати суму m останніх цифр числа N .	Дано дійсні числа x, ε . Обчислити з точністю ε нескінченну суму та вказати кількість урахованих доданків $\sum_{k=0}^{\infty} \frac{(-1)^k x^k}{(k+1)^2};$
29	Дано натуральне число N . а) З'ясувати, чи входить цифра 3 в запис числа N^2 . б) Поміняти порядок цифр числа N на зворотний.	Дано дійсні числа x, ε . Обчислити з точністю ε нескінченну суму та вказати кількість урахованих доданків $\sum_{k=0}^{\infty} \frac{x^{2k}}{2k!}$
30	Дано натуральне число N . а) Переставити першу і останню цифри числа N . б) Приписати по одиниці в початок і в кінець запису числа N .	Дано дійсні числа x, ε . Обчислити з точністю ε нескінченну суму та вказати кількість урахованих доданків $\sum_{k=0}^{\infty} \frac{x^{2k}}{2^k k!}$

2.4. Зміст звіту

1. Титульний лист з назвою комп'ютерного практикуму, номер варіанту, прізвище студента та групи.
2. Мета комп'ютерного практикуму.
3. Короткі теоретичні відомості.
4. Завдання.

5. Схеми алгоритмів.
6. Тексти програм.
7. Результати роботи програм.
8. Висновки про отримані результати роботи програм.

2.5. Контрольні питання

1. У чому відмінності між операторами *while* і *do while*?
2. Дайте поняття вкладених циклів?
3. Що таке цикл з передумовою?
4. Що таке цикл з постумовою?
5. Умова зупинки циклу *while*?
6. Для яких цілей використовуються цикли в програмуванні?
7. Перерахуйте оператори циклів в мові C.

Комп'ютерний практикум № 3. МАСИВИ

Мета комп'ютерного практикуму: вивчити базові операції з одновимірними і двовимірними масивами.

3.1. Теоретичні відомості

Представлення даних у вигляді окремих змінних не завжди достатньо при програмуванні реальних завдань. Наприклад, для представлення поведінки сигналу в часі або зберігання інформації про зображення зручно використовувати спеціальний тип даних - масиви. Одномірні масиви можна асоціювати з компонентами вектора, а двовимірні та з більшим ступенем вимірності - з матрицями. У загальному випадку масив - це набір елементів даних одного типу, для оголошення якого використовується наступний синтаксис (лінійний одновимірний масив):

<тип даних> <ім'я масиву> [кількість елементів];

Наприклад,

```
int array_int[100];    // одновимірний масив 100 цілочисельних
                       // елементів
double array_d[25];    // одновимірний масив 25 дійсних
                       // елементів
```

Як видно з прикладів, оголошення масивів відрізняється від оголошення звичайних змінних наявністю квадратних дужок []. Імена масивів вибираються за тими ж правилами, що й імена змінних. Звернення до окремого елементу масиву здійснюється за номером його індексу. У мові С індекси одного елементного виміру масиву можуть приймати значення від 0 до n-1. Це слід запам'ятати, інакше при виконанні програма почне звертатися до заборонених областей пам'яті з аварійним завершенням роботи. У мові С не має перевірки коректності значень індексу і відповідальність за це лежить на програмістові. Наступний фрагмент програми демонструє запис в масив значень лінійної функції $f(x) = kx + b$ і виведення цих значень на екран:

```
double k = 0.5, b = 10.0;
double f [100];
for (int x = 0; x < 100; x++) {
    f [x] = k * x + b;
    printf ("%f", f[x]);
}
```

У мові С передбачена можливість ініціалізації масиву в момент його оголошення, наприклад, таким чином:

```
int powers[4] = {1, 2, 4, 6};
```

У цьому випадку елементу `powers[0]` присвоюється значення 1, `powers[1]` - 2, і т.д. Особливістю ініціалізації масивів є те, що їх розмір можна задавати тільки константами, а не змінними. Наприклад, наступна програма призведе до помилки при компіляції:

```
int N = 100;
float array_f[N];    // помилка, так не можна
```

Тому при оголошенні масивів зазвичай використовують такий підхід:

```
#include <stdio.h>
#define N 100
int main () {
    float array_f[N];
    return 0;
}
```

Слід зазначити, що при ініціалізації масивів число їх елементів має збігатися з його розмірністю. Розглянемо варіант, коли число елементів при ініціалізації буде менше розмірності масиву:

```
#define SIZE 4
int data [SIZE] = {512, 1024};
for (int i = 0; i < SIZE; i++)
    printf ("%d, \n", data[i]);
```

Результат роботи програми буде наступним:

```
512, 1024, 0, 0
```

З отриманого результату видно, що неініціалізовані елементи масиву приймаються рівними нулю. У випадках, коли число елементів при ініціалізації перевищує розмірність масиву, то при компіляції відбудеться помилка. Тому, коли наперед невідомо число елементів, доцільно використовувати таку конструкцію мови C:

```
int data [] = {2, 16, 32, 64, 128, 256};
```

В результаті буде ініціалізовано одновимірний масив розмірністю 6 елементів. Тут залишається останнє питання: що буде, якщо значення індексу при зверненні до елементу масиву перевищить його розмірність? У цьому випадку ні програма, ні компілятор не видадуть повідомлень про помилку, але при цьому в програмі можуть виникати непередбачені помилки. Тому програмісту слід звертати особливу увагу на те, щоб індекси елементів при зверненні до елементів масиву не виходили за його межі. Також слід зазначити, що перший елемент масиву завжди має індекс 0, другий 1, і т.ін.

Для зберігання деяких видів інформації, наприклад, зображень зручно користуватися двовимірними масивами. Оголошення двовимірних масивів здійснюється наступним чином:

```
int array2D[100][20];           // двовимірний масив 100x20 елементів
```

Нумерація елементів також починається з нуля, тобто `array2D[0][0]` відповідає першому елементу першого рядка, `array2D[0][1]` – другому елементу першого рядка і т.ін. Для початкової ініціалізації двовимірного масиву може використовуватися наступна конструкція:

```
long array2D[3][2] = {{1, 2}, {3, 4}, {5, 6}};
```

або

```
long array2D[][] = {{1, 2}, {3, 4}, {5, 6}};
```

У загальному випадку можна задати масив будь-якої розмірності і правила роботи з ними аналогічні правилам роботи з одновимірними і двовимірними

масивами.

Приклад роботи з елементами двовимірного масиву за допомогою індексів:

```
#include <stdio.h>
#include <conio.h>
#define MAXSIZE 100

int main (void) {
    int i, /* лічильник рядків */
        j, /* лічильник стовпців */
        n, /* поточне число рядків у масиві */
        m, /* поточне число стовпців в масиві */
        num [MAXSIZE] [MAXSIZE]; /* сам двовимірний масив */

    printf ("Введіть число рядків і стовпців (не більше %d x %d):",
            MAXSIZE, MAXSIZE);
    scanf ("%d%d", &n, &m);
    if ((m>MAXSIZE) || (n>MAXSIZE)) {
        printf ("Перевищена допустима розмірність масиву!");
        getch (); /* затримка виконання програми! */
        return 1; /* повертаємо код помилки */
    }

    /* Заповнення масиву числами */
    for (i = 0; i < n; ++ i)
        for (j = 0; j < m; ++ j)
            num[i][j] = (i*n)+j+1; /* +1, так як індексація з нуля! */

    /* Друк масиву */
    for (j = 0; j < n; ++ j) {
        for (i = 0; i < m; ++ i)
            printf ("%3d", num[j][i]);
        printf ("\n"); /* перехід на новий рядок на екрані лише у кінці
                           поточного рядка масиву */
    }

    getch (); /* затримка виконання програми! */
    return 0; /* код нормального завершення програми */
}
```

3.2. Завдання на комп'ютерний практикум

1. Написати програму роботи з одновимірним масивом у відповідності з номером свого варіанта.
2. Написати програму роботи з двовимірним масивом у відповідності з номером свого варіанта.
3. Зробити висновки про отримані результати роботи програм.

3.3. Варіанти завдань

Варіант	Одновимірний масив	Двовимірний масив
1	Записати в масив значення функції $f(x) = kx + b$ при $x = \overline{1, N}$ і вивести його на екран. Значення k , b та N вводити з клавіатури	Занести в масив значення функції $f(x, y) = x + y$, $Xmin \leq x \leq Xmax$; $Ymin \leq y \leq Ymax$ і вивести його на екран. Значення $Xmin$, $Xmax$, $Ymin$ та $Ymax$ вводити з клавіатури
2	Записати в масив значення функції $f(x) = a \sin x/N$ при $x = \overline{1, N}$ і вивести його на екран. Значення a та N вводити з клавіатури	Написати програму вводу в масив розміром $N \times M$ дійсних чисел і пошуку в ньому максимального значення. Розміри і елементи матриці ввести з клавіатури
3	Написати програму вводу в масив N чисел і пошуку в ньому максимального значення. Значення N вводити з клавіатури	Занести в масив значення функції $f(x, y) = 1/(x + y)$ при $Xmin \leq x \leq Xmax$; $Ymin \leq y \leq Ymax$ і вивести його на екран. Значення $Xmin$, $Xmax$, $Ymin$ та $Ymax$ вводити з клавіатури
4	Записати в масив значення функції $f(x) = a \cos x/50$, при $x = \overline{1, N}$ і вивести його на екран. Значення a та N вводити з клавіатури	Написати програму вводу в масив розміром $N \times M$ цілих чисел і пошуку в ньому мінімального значення. Розміри і елементи матриці ввести з клавіатури
5	Написати програму вводу в масив N чисел і пошуку в ньому мінімального значення. Значення N вводити з клавіатури	Занести в масив значення функції $f(x, y) = (x + y)^2$ при $Xmin \leq x \leq Xmax$; $Ymin \leq y \leq Ymax$ і вивести його на екран. Значення $Xmin$, $Xmax$, $Ymin$ та $Ymax$ вводити з клавіатури
6	Записати в масив значення функції $f(x) = x^2 + b$, при $x = \overline{1, N}$ і вивести його на екран. Значення b та N вводити з клавіатури	Написати програму вводу в масив розміром $N \times M$ цілих чисел і обчислення суми елементів отриманого масиву. Розміри і елементи матриці ввести з клавіатури

Варіант	Одновимірний масив	Двовимірний масив
7	Написати програму вводу в масив N чисел і обчислення суми елементів отриманого масиву. Значення N вводити з клавіатури	Занести в масив значення функції $f(x, y) = 1 / ((x - y)^2 + 1)$ при $Xmin \leq x \leq Xmax; Ymin \leq y \leq Ymax$ і вивести його на екран. Значення $Xmin, Xmax, Ymin$ та $Ymax$ вводити з клавіатури
8	Написати програму вводу в масив N чисел і обчислення добутку парних від'ємних елементів отриманого масиву. Значення N вводити з клавіатури	Написати програму вводу в масив розміром $N \times N$ дійсних чисел і обчислення суми квадратів непарних елементів отриманого масиву. Розміри і елементи матриці ввести з клавіатури
9	Записати в масив значення функції $f(x) = 1/x + b$ при $x = \overline{1, N}$ і вивести його на екран. Значення N та b вводити з клавіатури	Занести в масив значення функції $f(x, y) = x - y$ при $Xmin \leq x \leq Xmax; Ymin \leq y \leq Ymax$ і вивести його на екран. Значення $Xmin, Xmax, Ymin$ та $Ymax$ вводити з клавіатури
10	Написати програму вводу в масив N чисел і пошуку в ньому максимального значення модуля. Значення N вводити з клавіатури	Дана матриця цілих чисел розміром $N \times M$. У рядках, всі елементи яких парні, розташувати елементи в зворотному порядку. Розміри і елементи матриці ввести з клавіатури.
11	Написати програму, яка перевіряє, чи являють елементи введеного з клавіатури масива із N чисел зростаючу послідовність. Значення N вводити з клавіатури	Написати програму, яка вводить в масив розміром $N \times N$ елементів дійсні числа і обчислює суму його елементів по стовбцям. Розміри і елементи матриці ввести з клавіатури

Варіант	Одновимірний масив	Двовимірний масив
12	Написати програму перевірки, чи є у введеному масиві з N елементів числа с однаковим значенням. Значення N вводити з клавіатури	Написати програму, яка вводить в масив розміром $N \times N$ дійсні числа і обчислює середнє арифметичне елементів за стовбцями. Розміри і елементи матриці ввести з клавіатури
13	Написати програму, яка обчислює, скільки разів введене з клавіатури число зустрічається в масиві з N елементів. Значення N вводити з клавіатури	Дана матриця символів $N \times M$. Підрахувати кількість рядків, в яких літер більше ніж цифр. Розміри і елементи матриці ввести з клавіатури
14	Написати програму, яка вводить з клавіатури зріст N студентів, обчислює середнє значення, а також кількість студентів, чий зріст вище середнього. Значення N вводити з клавіатури	Дана матриця розміром $N \times N$ дійсних чисел. Знайти максимальний елемент і найближчий до нього (за значенням) елемент матриці. Пошук здійснювати в квадратному контурі, центром якого є максимум, а довжина сторони - L елементів. Розмір матриці, її елементи та значення L вводити з клавіатури
15	Написати програму, яка обчислює середнє арифметичне ненульових елементів введеного з клавіатури масива з N цілих чисел. Значення N вводити з клавіатури	Написати програму, яка знаходить в матриці $N \times M$ номер рядка, сума елементів якого максимальна. Розмір матриці вводити з клавіатури
16	Написати програму, яка обчислює середнє арифметичне від'ємних елементів введеного з клавіатури масива з N цілих чисел. Значення N вводити з клавіатури	Написати програму, яка знаходить в матриці розміром $N \times M$ номер рядка, сума елементів якого мінімальна. Розмір матриці вводити з клавіатури. Розміри і елементи матриці ввести з клавіатури

Варіант	Одновимірний масив	Двовимірний масив
17	Дано одновимірний масив A , що складається з N елементів. Підрахувати максимальну кількість x позитивних чисел, що йдуть один за одним. Значення N вводити з клавіатури	Написати програму, яка знаходить в матриці розміром $N \times M$ номер рядка, сума модулів елементів якого максимальна. Розміри і елементи матриці ввести з клавіатури
18	Написати програму, яка перевіряє, чи являють елементи введеного з клавіатури масива з N чисел спадаючу послідовність. Значення N вводити з клавіатури	Дана матриця дійсних чисел. Знайти мінімальний елемент і найближчий до нього (за значенням) елемент матриці. Пошук здійснювати у напрямі, паралельному головній діагоналі. Розміри і елементи матриці ввести з клавіатури
19	Дан масив чисел $a_1, \dots, a_N$. З'ясувати, чи є в даному масиві два позитивних елемента, що йдуть підряд. Підрахувати кількість таких пар. Значення N вводити з клавіатури	Написати програму, яка знаходить в матриці $N \times M$ номер рядка, середнє арифметичне якого мінімальне. Розміри і елементи матриці ввести з клавіатури
20	Написати програму, яка перевіряє, чи являють елементи введеного з клавіатури масива з N чисел геометричну прогресію. Значення N вводити з клавіатури	Дана матриця дійсних чисел $N \times M$. Знайти серед елементів матриці число з максимальною дробовою частиною, переставити рядки і стовпці матриці так, щоб це число стояло в верхньому лівому кутку матриці. Розміри і елементи матриці ввести з клавіатури
21	У масиві дійсних чисел поміняти місцями максимальний від'ємний елемент і мінімальний додатній. Кількість елементів в масиві не більше двадцяти. Елементи і їх кількість ввести з клавіатури.	Написати програму, яка вводить в масив розміром $N \times M$ дійсні числа і обчислює суму його елементів за рядками. Розміри і елементи матриці ввести з клавіатури

Варіант	Одновимірний масив	Двовимірний масив
22	У масиві дійсних чисел серед чисел, що стоять на парних місцях, визначити мінімальний додатній елемент масиву і його індекс. Кількість елементів в масиві не більше двадцяти. Елементи і їх кількість ввести	Ввести матрицю чисел $N \times N$ і транспонувати її. Визначити рядки, в яких значення елементів зменшуються, і стовпці, в яких вони зростають. Розміри і елементи матриці ввести з клавіатури
23	У масиві цілих чисел серед чисел, що стоять на непарних місцях, визначити максимальний від'ємний елемент масиву і його індекс. Кількість елементів в масиві не більше двадцяти. Елементи і їх кількість ввести з клавіатури	Дана матриця $N \times N$. Обчислити суму тих її елементів, які розташовані на головній діагоналі і вище, та які при цьому перевершують за значенням всі елементи, розташовані нижче головної діагоналі. Якщо таких елементів немає, то видати відповідне повідомлення. Розміри і елементи матриці ввести з клавіатури
24	У масиві дійсних чисел поміняти місцями мінімальний від'ємний і максимальний додатній елементи. Кількість елементів в масиві не більше двадцяти п'яти. Елементи і їх кількість ввести	Написати програму, яка виводить значення елементів матриці $N \times N$, утвореної з вихідної матриці, шляхом видалення всіх рядків і стовпців, що містять хоча б один нульовий елемент. Розміри і елементи матриці ввести з клавіатури
25	Визначити суму елементів, що мають непарні значення і що стоять на парних місцях. Кількість елементів в масиві не більше п'ятнадцяти. Елементи і їх кількість ввести	Дана дійсна квадратна матриця розмірністю $N \times N$. Знайти і вивести на екран: а) індекси усіх її ненульових елементів; б) кількість усіх її різних елементів Розміри і елементи матриці ввести з клавіатури

Варіант	Одновимірний масив	Двовимірний масив
26	У масиві цілих чисел визначити суму елементів, що стоять на парних позиціях, і добуток елементів, що мають непарні значення. Кількість елементів в масиві не більше двадцяти. Елементи і їх кількість ввести	Визначити суму елементів, розташованих в парних стовбцях матриці нижче бічної діагоналі. Розмірність матриці не більше $N \times N$. Розмірність і елементи матриці ввести з клавіатури. Елементи - цілі числа
27	У масиві визначити кількість елементів, кратних максимальному елементу. Кількість елементів в масиві не більше двадцяти. Елементи і їх кількість ввести з клавіатури	Елемент матриці розміром $N \times M$ дійсних чисел називається <i>особливим</i> , якщо: 1) він більший за суму інших елементів свого стовпчика; 2) у його рядку ліворуч від нього всі розташовані елементи менші за нього, а праворуч – більші. Знайти і вивести кількість <i>особливих</i> елементів матриці та їх індекси. Розмірність і елементи матриці ввести з клавіатури
28	У масиві цілих чисел вивести значення кожного елементу, що стоїть після елементу зі значенням, кратним п'яти. Розмірність масиву і його елементи ввести з клавіатури	Визначити максимальний елемент матриці дійсних чисел розмірністю $N \times M$ і його індекси в кожному рядку матриці. Розмірність матриці не більше 30×10 . Розміри і елементи матриці ввести з клавіатури
29	У масиві дійсних чисел визначити п'ять максимальних елементів. Розмірність масиву і числа ввести з клавіатури	Визначити максимальний елемент і його позицію серед елементів, розташованих вище головної діагоналі матриці цілих чисел. Розміри матриці не більше 10×10 елементів. Поточні розміри і елементи матриці ввести з клавіатури

Варіант	Одновимірний масив	Двовимірний масив
30	У масиві цілих чисел замінити нулями від'ємні елементи, які стоять зразу після додатних. Елементів в масиві не більше двадцяти. Розмірність масиву і його елементи ввести з клавіатури	Поміняти місцями максимальний елемент серед всіх від'ємних елементів матриці дійсних чисел на мінімальний елемент серед усіх додатних. Розмірність матриці дійсних чисел не більше 20x20. Поточний розмір і елементи матриці ввести з клавіатури

3.4. Зміст звіту

1. Титульний лист з назвою комп'ютерного практикуму, номер варіанту, прізвище студента та групи.
2. Мета комп'ютерного практикуму.
3. Короткі теоретичні відомості.
4. Завдання.
5. Схеми алгоритмів.
6. Тексти програм.
7. Результати роботи програм.
8. Висновки про отримані результати роботи програм.

3.5. Контрольні питання

1. Яким чином задаються одномірні масиви в мові C?
2. Запишіть масив цілих чисел з початковими значеннями 1, 2 і 3.
3. Яким чином задаються двовимірні масиви в мові C?
4. У чому переваги масивів перед змінними?
5. Як записати значення в елемент масиву?
6. Як відобразити елементи масиву на екрані монітору?

Комп'ютерний практикум № 4. РОБОТА З РЯДКАМИ В МОВІ С

Мета комп'ютерного практикуму: вивчити базові операції роботи з рядками.

4.1. Теоретичні відомості

У мові ANSI C89 немає спеціального типу даних для рядкових змінних. Для цих цілей використовуються масиви символів (тип **char**). Наступний приклад демонструє використання рядків у програмі:

```
char str_1[100] = {'П', 'р', 'у', 'в', 'і', 'м', '\0'};  
char str_2[100] = "Привіт";  
char str_3[] = "Привіт";  
printf ("%s\n%s\n%s\n", str_1, str_2, str_3);
```

У наведеному прикладі показані три способи ініціалізації рядкових змінних. Перший спосіб є класичним оголошенням масиву, другий і третій використовуються спеціально для рядків. Причому в останньому випадку, компілятор сам визначає потрібну довжину масиву для запису рядка. Аналізуючи перший і другий способи ініціалізації масиву символів виникає питання: яким чином мова С «знає» де закінчується рядок? Дійсно, масив *str_2* містить 100 елементів, а масив *str_3* менше 100, проте довжина рядка і в першому і в другому випадках одна і та ж. Такий ефект досягається за рахунок використання спеціальних керуючих кодів, які говорять де закінчується рядок або де використовується перенос усередині одного рядка і т.п. Зокрема символ '\0' означає в мові С кінець рядка і всі символи після нього ігноруються як символи рядка. Наступний приклад показує особливість використання даного спеціального символу.

```
char str1[10] = {'Л', 'е', 'к', 'ц', 'і', 'я', '\0'};  
char str2[10] = {'Л', 'е', 'к', 'ц', '\0', 'і', 'я'};  
char str3[10] = {'Л', 'е', '\0', 'к', 'ц', 'і', 'я'};  
printf ("%s\n%s\n%s\n", str1, str2, str3);
```

Результатом роботи даного коду буде друк наступних трьох рядків:

```
лекція  
лекц  
ле
```

З цього прикладу видно як символ кінця рядка '\0' впливає на довжину рядків. Таким чином, щоб підрахувати довжину рядка (число символів) необхідно рахувати символи доти, поки не зустрінеться символ '\0' або не буде досягнутий кінець масиву (у цьому випадку слід розуміти, що ми стикнулися з масивом символів, а не з рядком мови С). Функція обчислення розміру рядків уже реалізована в стандартній бібліотеці мови С, яку можна підключати до програми за допомогою заголовкового файлу **string.h**, і має наступний синтаксис:

```
int strlen (char* str);
```

де *char* str* - покажчик на рядок (покажчикам присвячений комп'ютерний практикум №7). Наступна програма показує використання функції **strlen()**.

Лістинг 4.1. Приклад використання функції **strlen()**.

```
#include <stdio.h>
#include <string.h>

int main (void) {
    char str[] = "Привіт свім!";
    int length = strlen(str);
    printf ("Довжина рядка =%d. \n", length);
    return 0;
}
```

Результатом роботи програми буде виведення на екран числа 12. Враховуючи, що перший символ рядка має нульовий індекс, то можна помітити, що дана програма рахує усі символи рядка аж до символу закінчення рядка '\0'. **Тому індекс останнього елемента в масиві рядка буде 11, а не 12!**

Тепер розглянемо правила присвоювання однієї строкової змінної іншій. Припустимо, оголошені два рядки

```
char str1[] = "Це перший рядок";
char str2[] = "Це другий рядок";
```

і необхідно виконати оператор присвоювання

```
str1 = str2;
```

При такому записі оператора присвоювання компілятор видасть повідомлення про помилку. Для того, щоб виконати присвоювання слід виконати посимвольне копіювання кожного елемента масива *str2* на відповідне місце в масиві *str1*. Функція копіювання вже реалізована в бібліотеці мови C (заголовковий файл **string.h**) і має наступний вигляд:

```
char* strcpy (char* dest, char* src);
```

Вона виконує копіювання рядка *src* в рядок *dest* і повертає готовий рядок *dest*. У лістингу 4.2 показано використання функції **strcpy()**.

Лістинг 4.2. Приклад використання функції strcpy().

```
#include <stdio.h>
#include <string.h>
int main (void) {
    char src[] = "Привіт свім!";
    char dest[100];
    strcpy (dest, src);
    printf ("%s\n", dest);
    return 0;
}
```

Крім операцій обчислення довжини рядка і копіювання рядків важливою є операція порівняння двох рядків між собою. У мові C два рядки вважаються однаковими, якщо довжини і елементи одного рядка дорівнюють відповідним елементам іншого. Функція порівняння двох рядків має вигляд:

```
int strcmp (char *str1, char *str2);
```

і реалізована в бібліотеці **string.h**. Ця функція повертає нуль, якщо рядки *str1* і *str2* рівні і не нуль в іншому випадку (-1, якщо перший рядок менший за другий і +1, якщо навпаки). Наведемо приклад використання даної функції:

```
char str1[] = "Це перший рядок";  
char str2[] = "Це другий рядок";  
if (!strcmp (str1, str2))  
    printf ("Рядок %s дорівнює рядку %s\n", str1, str2);  
else  
    printf ("Рядок %s не дорівнює рядку %s\n", str1, str2);
```

У мові C є кілька функцій, що дозволяють вводити рядки з клавіатури. Найпоширенішою з них є раніше розглянута функція **scanf()**, якою як параметр передається посилання на масив символів:

```
char str[100];  
scanf ("%s", str);
```

В результаті виконання цього коду, змінна **str** буде містити введену користувачем послідовність символів (слід мати на увазі, що специфікатор %s буде вводити рядок лише до першого символа-розділювача слів в ньому). Крім функції **scanf()**, для учбових завдань в літературі часто використовують функцію **gets()** бібліотеки **stdio.h**, яка в якості аргументу приймає посилання на масив символів:

```
gets(str);
```

Ця функція зчитує символи доти, поки користувач не натисне клавішу *Enter*, тобто введе символ переведення рядка '\n'. Потім вона записує замість символу '\n' символ '\0' і передає рядок програмі, яка її викликає. Оскільки функція **gets()** не контролює розмір рядка, що вводиться, її використання в програмах часто призводить до аварійного завершення програми через перевищення розміру масиву рядка *str* при введенні. Для уникнення цієї помилки у програмах, написаних в рамках даних лабораторних робіт і в реальному житті слід використовувати лише функції **fgets()** чи **s_gets()**, які дозволяють контролювати розмір введеного рядка.

Для виведення рядків на екран крім функції **printf()** можна використовувати також функцію **puts()** бібліотеки **stdio.h**, яка простіша у використанні і швидша в роботі. Наступний приклад демонструє застосування даної функції:

```
#define DEF "Заданий рядок"  
char str[] = "Це перший рядок";  
puts (str);  
puts (DEF);  
puts (&str[4]);
```

Результат роботи цього коду наступний:

Це перший рядок

Заданий рядок
перший рядок

Ще однією зручною функцією мови C для роботи з рядками є функція **sprintf()** бібліотеки **stdio.h**. Її дія аналогічна розглянутій раніше функції **printf()** з однією лише різницею, що результат виводу не друкується на екрані, а заноситься в строкову змінну у пам'яті:

```
int age;
char name[100], str[100];
printf ("Введіть Ваше ім'я:");
scanf ("%s", name);
printf ("Введіть Ваш вік:");
scanf ("%d", & age);
sprintf (str, "Добрий день %s. Ваш вік %d років", name, age);
puts (str);
```

В результаті масив *str* буде містити рядок «Добрий день <такий-то>. Ваш вік <стільки-то років>».

Аналіз останнього прикладу показує, що за допомогою функції *sprintf()* можна перетворювати числові змінні в рядкові, об'єднувати кілька рядків в один і т.ін.

4.2. Завдання на комп'ютерний практикум

1. Написати дві програми по роботі з рядками у відповідності до номеру свого варіанту.
2. У завданнях, де не сказано інше, уся вхідна інформація для програми повинна вводитися з клавіатури.

4.3. Варіанти завдань

Варіант	1 завдання	2 завдання
1	Написати програму поелементного копіювання довільного рядка в інший рядок з викресленням кожної парної літери	Написати програму об'єднання трьох довільних рядків в четвертий рядок без використання функцій та операцій мови C
2	Написати програму заміни у введеному рядку малих літер на великі, не використовуючи стандартні функції мови C	Написати програму видалення літери <i>ch</i> з введеного рядка. Літеру <i>ch</i> задати з клавіатури
3	Написати програму підрахунку літери <i>ch</i> у введеному рядку. Літеру <i>ch</i> задати з клавіатури	Написати програму додавання слова <i>word</i> після першого слова введеного рядка* Слово <i>word</i> задати з клавіатури

Варіант	1 завдання	2 завдання
4	Написати програму видалення літери <i>ch</i> з останнього слова введеного рядка. Літеру <i>ch</i> задати з клавіатури	Написати програму лекскографічного порівняння двох рядків, не використовуючи стандартних функцій мови C
5	Написати програму додавання символу <i>ch1</i> після кожного символу <i>ch2</i> введеного рядка. Літери <i>ch1</i> та <i>ch2</i> слід ввести з клавіатури	Написати програму заміни у введеному рядку заголовних літер <i>ch</i> на прописні. Літеру <i>ch</i> слід ввести з клавіатури
6	Написати програму підрахунку числа слів у рядку*	Написати програму підрахунку числа літер <i>ch</i> у введеному рядку. Літеру <i>ch</i> слід ввести з клавіатури
7	Написати програму виділення першого слова з введеного рядка* і відображення його на екрані	Написати програму видалення всіх символів <i>ch</i> з введеного рядка. Символ <i>ch</i> слід ввести з клавіатури
8	Написати програму виділення останнього слова з введеного рядка* і відображення його на екрані	Написати програму копіювання першої половини введеного рядка в інший рядок
9	Написати програму виведення кожного слова в рядку задом наперед.	Написати програму лексикографічного порівняння перших половин двох введених рядків
10	Написати програму видалення останнього слова з рядка*	Написати програму заміщення першої половини рядку другою, а другої – першою.
11	Написати програму, яка видаляє із введеного з клавіатури рядка початкові пробіли	Дано рядок, слова в ньому розділені пробілом. Поміняти місцями парні і непарні по порядку проходження слова.
12	Написати програму, яка у рядку переводить наступний символ після крапки у верхній регістр. Не використовувати для цього стандартні функції мови C.	Написати програму, яка знаходить в рядку слово під заданим номером и виводить його на екран*

Варіант	1 завдання	2 завдання
13	Написати програму поелементного копіювання довільного введеного рядка в інший символний масив з викреслюванням голосних літер	Написати програму, яка замінює останні N символів слів в рядку, що мають довжину L більше введеної з клавіатури, на символи "?"* L та N задаються з клавіатури
14	Написати програму, яка переписує у вихідний рядок слова із вхідного рядка в порядку зростання їх довжини*	Написати програму визначення співвідношення строкових і прописних літер у введеному рядку
15	Написати програму заміни послідовності довільних однакових символів (не пробілів) на десяткове число, що відповідає їх кількості, і сам символ (тобто "abcdaaaaa xyznnnnnn" на "abcd5a xyz6n")	Написати програму видалення з рядка всіх слів, довжина яких менша за L^* . L задати з клавіатури
16	Написати програму заміни в кожному введеному слові останнього символу на цей же символ у верхньому регістрі*. Не використовувати для цього стандартної функції мови C	Написати програму, яка вставляє в задану позицію першого рядка другий рядок
17	Написати програму заміни символів у рядку їх ASCII кодами. Коди виводити через пробіл	Написати програму, яка видаляє із рядка задану кількість символів, починаючи з заданої позиції
18	Написати програму заміни символа $ch1$ на $ch2$ у найдовшому слові рядка*	Написати програму, яка додає в рядок пробіл після розділових знаків (.,:!?), якщо він там відсутній
19	Написати програму переводу символів у найкоротшому слові рядка у верхній регістр*. Не використовувати для цього стандартної функції мови C	Написати програму видалення з рядку слів, що повторюються*
20	Написати програму визначення, чи є введений рядок паліндромом	Написати програму, яка визначає, як часто той або інший символ рядка зустрічається у даному рядку

Варіант	1 завдання	2 завдання
21	Дано речення, слова в ньому розділені пробілом, поміняти місцями перше й останнє слово	Написати програму, яка визначає позицію першого (останнього) входження заданого символу до заданого рядка. Функцію <i>strchr()</i> не використовувати
22	Дан текст. Якщо в ньому немає малих латинських літер, то залишити його без змін, в іншому випадку кожен з символів, які йдуть за першою групою малих латинських літер, замінити крапкою	У реченні знайти всі однокореневі слова. Корінь та речення задаються з клавіатури
23	Дано речення, слова в ньому розділені пробілом, поміняти місцями перше й останнє слово	Видалити в реченні всі повторні входження слів і роздрукувати його заново
24	Дано речення, слова в ньому розділені пробілом, поміняти місцями парні і непарні по порядку проходження слова речення	Рядок містить арифметичний вираз, у якому використовуються круглі дужки. Перевірити, чи правильно в ньому стоять дужки (кожна відкрита дужка повинна мати пару – закриту дужку)
25	Дано речення, слова в ньому розділені пробілом. Підрахувати кількість слів, які починаються з тієї букви, якою закінчується попереднє слово	Написати програму, що повертає кількість різних символів у рядку
26	Дано речення, слова в ньому розділені пробілом. Підрахувати кількість слів, які починаються з тієї ж букви, що і наступне слово	Дано слово з 12 літер. Поміняти місцями його третини наступним чином: першу третину слова розмістити на місці останньої, середню третину - на місці першої, останню третину - на місці середньої

Варіант	1 завдання	2 завдання
27	Дано речення, слова в ньому розділені пробілом. Впорядкувати слова в реченні в порядку зростання їх довжини	Знайти символ, який входить до рядка <i>S</i> найбільшу кількість разів. Вказівка. Визначити масив з індексами символного типу та елементами натурального типу. Обчислити та зберегти кількість входжень кожного символу до рядка <i>S</i> . Знайти максимальний елемент цього масиву
28	Дано речення, слова в ньому розділені пробілом. Впорядкувати слова за абеткою (тільки по першій літері)	Визначити, чи є заданий текст правильним записом цілого числа, чи ні.
29	Дано речення, слова в ньому розділені пробілом. Створити з нього два рядки символів по правилу: в перший рядок переписати всі парні по порядку проходження слова, а в другий - непарні	Дано число n із проміжку від 0 до 10^9 . Надрукувати його українськими словами (наприклад, 234 = двісті тридцять чотири)
30	Дано речення. Роздрукувати з нього всі слова, перед якими безпосередньо знаходиться літера <i>ch</i> в одному слові. Літера <i>ch</i> вводиться з клавіатури	Написати програму, яка всі цифри, що містяться у введеному рядку, замінює їх квадратами і виводить на екран заново

* вважається, що слова розділяються пробілом

4.4. Зміст звіту

1. Титульний лист з назвою комп'ютерного практикуму, номером варіанту, прізвищем студента та групи.
2. Мета комп'ютерного практикуму.
3. Короткі теоретичні відомості.
4. Завдання.
5. Схеми алгоритмів.
6. Тексти програм.
7. Результати роботи програм.
8. Висновки про отримані результати роботи програм.

4.5. Контрольні запитання

1. Як задаються рядки в програмі на мові C?
2. Для чого призначена функція *strcpy()* і в якій бібліотеці вона визначена?

3. Запишіть можливі способи початкової ініціалізації рядка.
4. Який керуючий символ відповідає кінцю рядка?
5. Що виконує функція *strcmp()*?
6. Яку роль відіграють структури в програмуванні?
7. Що повертає функція *strlen()*?

Комп'ютерний практикум № 5. ПОКАЖЧИКИ

Мета комп'ютерного практикуму: ознайомитися з механізмом динамічного виділення пам'яті і роботою з покажчиками.

5.1. Теоретичні відомості

Поняття покажчика

Будь-який об'єкт програми, чи то звичайна змінна базового або користувацького типу, або масив, займає в пам'яті певну область. Місце розташування об'єкта в пам'яті визначається його адресою. Адреса показує, де саме в пам'яті програми (з якого по рахунку байта пам'яті) розміщується значення змінної.

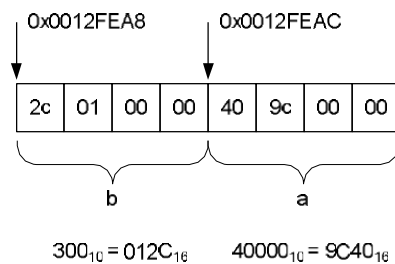


Рисунок 5.1. Представлення змінних *a* і *b* в пам'яті

Щоб дізнатися адресу конкретної змінної, використовується унарна операція взяття адреси. Для застосування даної операції до об'єкта слід вказати знак амперсанда перед його ім'ям (&).

Мова C дозволяє здійснювати доступ до пам'яті за допомогою спеціальних змінних, які називаються покажчиками. Покажчик – це змінна, що містить адресу деякої комірки пам'яті. Покажчик визначає не тільки адресу, але і тип даних об'єкта, що розміщується в пам'яті за заданою адресою.

При оголошенні покажчика використовується наступний синтаксис:

```
тип_даних* ім'я_покажчика
```

Від оголошення звичайної змінної дана конструкція відрізняється наявністю символу «зірочка» (*), який визначає, що новий ідентифікатор буде покажчиком. Під покажчик, як і під будь-яку іншу змінну, відводиться місце в пам'яті (чотири байти), причому розмір покажчика не залежить від типу даних значення, на яке він посилається.

Для запису певної адреси в покажчик використовується оператор присвоювання. Покажчик, як і будь-яка інша змінна, може бути ініціалізованим відразу при своєму оголошенні.

```
// Оголошуємо символьну змінну Symbol
char Symbol = 'Y';
// Оголошуємо показник на змінну Symbol
char* pSymbol = & Symbol;
// Оголошуємо цілочисельну змінну Value
int Value = 1234;
// Оголошуємо показник для посилання на
// цілочисельні змінні ...
int* pValue;
// ... і присвоюємо йому адресу змінної Value
pValue = & Value;
```

Показник може бути оголошений з типом даних **void**. Вказівник **void*** використовують у випадку, коли тип даних об'єкта, на який буде посилатися показник, заздалегідь невідомий. Надалі, **до використання (!)** такий показник повинен бути явно приведений до якого-небудь типу даних.

Операції з показниками

До показників може бути застосована спеціальна операція розіменування. **Операція розіменування** дозволяє звернутися до значення, яке розміщується в пам'яті за заданою адресою. Для позначення даної операції використовується символ «зірочка» (*). «Зірочка» ставиться перед показником, через який виконується звернення до значення в пам'яті.

Операція розіменування повертає значення відповідно до типу даних показника. Слід зазначити, що операцію розіменування не можна застосовувати до нетипізованих показників **void*** без їх приведення до певного типу.

Операція розіменування може використовуватися не тільки для читання, але й для зміни даних, що розміщуються в пам'яті.

```
// Оголошуємо цілу змінну Value1:
int Value1 = 10;
// Оголошуємо показник pValue1
// і присвоюємо йому адресу змінної Value1
int* pValue1 = &Value1;
// Оголошуємо цілу змінну Value2
// і присвоюємо їй значення, що зберігається за
// адресою pValue1 (значення змінної Value1):
int Value2 = *pValue1;
```

До показників також можуть застосовуватися деякі арифметичні операції і операції порівняння (див. таблицю нижче). Крім того, до показників можна застосовувати оператори інкремента і декремента.

Операції з показниками

Операція	Вираз	Результат	Опис
Дорівнює (==)	показчик1 == показчик2	true/ false	Порівнює два показника на рівність адрес
Не дорівнює (!=)	показчик1 != показчик2	true/ false	Порівнює два показника на нерівність адрес

Операція	Вираз	Результат	Опис
Меньше (<)	показчик1 < показчик2	true/ false	Повертає істину, якщо адреса <i>показчика1</i> менша адреси <i>показчика2</i>
Меньше або дорівнює (<=)	показчик1 <= показчик2	true/ false	Повертає істину, якщо адреса показчика1 менша або дорівнює адресі показчика2
Більше (>)	показчик1 > показчик2	true/ false	Повертає істину, якщо адреса показчика1 більша адреси показчика2
Більше або дорівнює (>=)	показчик1 >= показчик2	true/ false	Повертає істину, якщо адреса показчика1 більша або дорівнює адресі показчика2
Віднімання(-)	показчик1 – показчик2	ціле число	Обчислює кількість елементів заданого типу між показчиками
Віднімання(-)	показчик – ціле_число	показчик	Обчислює показчик, віддалений від заданого на певну кількість елементів (відповідно до типу показчика)
Додавання(+)	показчик + ціле_число	показчик	
Інкремент (++)	показчик++, чи ++показчик	показчик	Обчислює показчик, віддалений від заданого на 1 елемент
Декремент (--)	показчик--, чи --показчик	показчик	

Операції порівняння дозволяють перевірити рівність/нерівність адрес, що містяться у показчиках, а також визначити взаємне розташування значень у пам'яті.

Лістинг 5.1:

```
// Переконаємося, що показчики ptr1 і ptr2
// посилаються на одну й ту саму ділянку пам'яті
int a = 1;
int* ptr1 = &a;
int* ptr2 = &a;
if (ptr1 == ptr2)
    printf ("Вказівники посилаються
            на одну змінну");
```

Лістинг 5.2:

```
// Переконаємося, що покажчики ptr1 і ptr2
// посилаються на різні області пам'яті
int a = 1;
int b = 1;
int* ptr1 = &a;
int* ptr2 = &b;
if (ptr1 != ptr2)
    printf ("Вказівники посилаються
           на різні змінні ");
```

Лістинг 5.3:

```
// Визначимо, в якому порядку
// в пам'яті розміщуються змінні a і b
int a = 1;
int b = 1;
int* ptr1 = &a;
int* ptr2 = &b;
printf ("%p\n%p", ptr1, ptr2);
if (ptr1 < ptr2)
    printf ("ptr1 < ptr2");
```

В результаті додавання до покажчика цілого числа N виходить нова адреса, яка більша за вихідну на M байт, причому

$M = N * \text{розмір_типу_даних_вказівника}$.

Наприклад, при додаванні до покажчика типу *double** цілого числа 2, ми змістимося в пам'яті на 16 байт ($16 = 2 * 8$, 8 - розмір типу даних *double*). Аналогічним чином працює операція віднімання (цілого числа) і оператори інкремента/декремента.

Використання арифметичних операцій, а також операторів інкремента/декремента дозволяє пересуватися в пам'яті від одного елемента масиву до іншого, наприклад:

```
double X[]={1.1, 2.5, 3.7, 4, 5};
double* ptr = &(X[0]);
printf ("%p%d\n",ptr,*ptr);
ptr++;
printf ("%p%d\n",ptr,*ptr);
ptr = ptr+2;
printf ("%p%d\n",ptr,*ptr);
ptr = ptr-3;
printf ("%p%d\n",ptr,*ptr);
```

Результати:

```
002EF794 1.1
002EF79C 2.5
002EF7AC 4
002EF794 1.1
```

Операція віднімання покажчика від покажчика може застосовуватися лише в тому випадку, якщо обидва покажчики мають однаковий тип даних. Результатом віднімання значення покажчика від покажчика є ціле число N , що визначає кількість елементів заданого типу, які розміщуються в пам'яті між двома адресами. Число N визначається як відношення кількості байт, що розділяють в пам'яті дві адреси, до розміру типу даних покажчика.

Особливості застосування покажчиків

Покажчик може містити спеціальне значення *NULL*. Значення *NULL* відповідає нульовій адресі і говорить про те, що покажчик не посилається на яку-небудь комірку пам'яті (іншими словами, ні на що не вказує).

Слід зазначити, що спроба розіменувати нульовий покажчик призведе до аварійної зупинки програми:

```
int* A = NULL;
printf ("%d", *A); // !!! Помилка (Access
                  // Violation)
                  // Розіменувати адресу NULL
                  // не можна
```

Ім'я масиву, який оголошується, завжди асоціюється компілятором з адресою його першого елемента. Таким чином, ідентифікатор, що позначає ім'я масиву, одночасно є і покажчиком на перший елемент цього масиву.

Приклад:

```
int A[6] = {0,1,2,3,4,5};
printf ("%p", A);
```

Можливий результат:

```
0012FE98
```

Розіменовуючи ідентифікатор масиву, ми отримуємо доступ до елемента з індексом 0:

```
int A[5] = {10,20,30,40,50};
printf ("%d", *A); // printf ("%d", *A[0]);
```

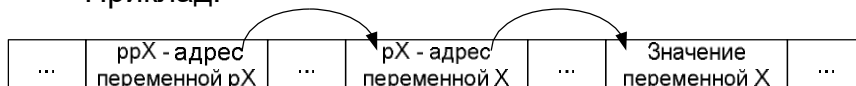
Додаючи до покажчика цілі числа, можна переміщуватись по об'єктах масиву, як з використанням індексів:

```
int A[5] = {10,20,30,40,50};
printf ("%d", *(A+2)); // printf ("%d", *A[2]);
```

Покажчики на покажчики

Покажчик може посилатися на змінну, яка, в свою чергу, також є покажчиком. У такому випадку, в комірку пам'яті, на яку посилається покажчик в якості значення буде міститися адреса іншої комірки пам'яті.

Приклад:



```
int X = 10;
int* pX = &X;
int** ppX = &pX;
printf ("%i%p%p\n", X, pX, ppX);
printf ("%i%p%p\n", **ppX, *ppX, ppX);
```

Можливий результат:

```
10 002EFC38 002EFC2C
10 002EFC38 002EFC2C
```

Типи програмної пам'яті

У 32-бітних системах процесор використовує для адресації пам'яті 4 байтні значення. Це означає, що кожному процесу (кожній програмі) виділяється діапазон віртуальних адрес від 0x00000000 до 0xFFFFFFFF - всього 4 Гб адресного простору.

За замовчуванням адресний простір кожного процесу ізолювано. Дані двох різних процесів, записані за однією і тією ж віртуальною адресою, виявляються в різних сторінках фізичної пам'яті за допомогою коректної роботи системи трансляції адреси.

В адресному просторі під програму відводиться декілька областей, серед них:

- **область коду (code segment)** - містить виконувані машинні коди функцій програми

- **область даних (data segment)** - містить глобальні та статичні змінні, а також рядкові константи

- **стек (stack)** - містить локальні змінні, параметри виклику функцій, а також ряд інших даних.

- **«купа» (heap)** - використовується для зберігання динамічних даних.

Для кожної змінної відводиться певне місце в пам'яті. При цьому пам'ять під змінні може виділятися статично і динамічно.

Статичне виділення пам'яті здійснюється для всіх оголошених у програмі змінних. При цьому задіюються лише область даних і стек. При статичному виділенні пам'яті заздалегідь відомо, скільки пам'яті буде використовуватися для зберігання даних.

Пам'ять під змінні може виділятися **динамічно**, тобто в процесі роботи програми. Змінні, для яких пам'ять була виділена в процесі роботи програми, зазвичай називають **динамічними змінними**.

Динамічні змінні створюються за допомогою спеціальних функцій або операторів і розміщуються в області пам'яті «купа». Вони існують до тих пір, поки не будуть явно знищені також за допомогою спеціальних функцій або операторів. Динамічні змінні не мають імені. Звернення до них здійснюється лише через їх адреси за допомогою покажчиків.

Динамічне виділення пам'яті, часто використовується для масивів. Як ви знаєте, при оголошенні масиву, необхідно вказати його розмір. Однак розмір масиву може бути не відомий заздалегідь, до виконання програми. У такому випадку, можна виділити пам'ять під масив динамічно, після того, як розмір масиву стане відомим.

Керування динамічною пам'яттю

Пам'ять, яка виділяється в Сі функціями динамічного розподілу даних, знаходиться в так званій динамічно розподіленій області пам'яті "купа". Динамічно розподілена область пам'яті - це вільна область пам'яті, що не використовується програмою, операційною системою або іншими програмами. Розмір динамічно розподіленої області пам'яті заздалегідь невідомий, але як правило в ній досить пам'яті для розміщення даних програми. Більшість компіляторів підтримують бібліотечні функції, що дозволяють отримати поточний розмір динамічно розподіленої області пам'яті. Хоча розмір динамічно розподіленої області пам'яті дуже великий, все ж вона скінченна і

може бути вичерпана. Тому при створенні нового динамічного об'єкта обов'язковою умовою є перевірка результату створення до першого використання цього об'єкта.

Основу системи динамічного розподілу в ANSI C89 складають функції *malloc()* і *free()*. Ці функції працюють спільно. Їх прототипи знаходяться в заголовному файлі *stdlib.h*. Функція *malloc()* виділяє пам'ять, а *free()* - звільняє її. Це означає, що при кожному запиті функція *malloc()* виділяє необхідну ділянку вільної пам'яті, а *free()* - звільняє її, тобто повертає системі. В програму, яка використовує ці функції, потрібно включити заголовний файл *<stdlib.h>*.

Прототип функції *malloc()* наступний:

```
void* malloc(size_t кількість_байтів);
```

Тут *кількість_байтів* – це загальний розмір пам'яті в байтах, необхідний для розміщення усіх нових даних. (Тип *size_t* визначено в *<stdlib.h>* як деякий цілий тип без знаку.) Функція *malloc()* завжди повертає покажчик типу *void**, тому його можна присвоювати покажчику будь-якого типу. При успішному виконанні *malloc()* повертає покажчик на перший байт неперервної ділянки пам'яті, виділеної в динамічно розподіленій області пам'яті. Якщо в динамічно розподіленій області пам'яті недостатньо вільної пам'яті для виконання запиту, то пам'ять не виділяється і *malloc()* повертає нуль (*NULL*).

Якщо ж при виділенні нової області пам'яті потрібно проініціалізувати її нулями - слід використовувати функцію *calloc()*. Її прототип виглядає так:

```
void* calloc(size_t num, size_t size);
```

Функція *calloc()* виділяє пам'ять, розмір якої дорівнює значенню добутку *num*size*, тобто пам'ять, достатню для розміщення масиву, що містить *num* об'єктів розміром *size*. Всі байти виділеної пам'яті ініціалізуються нулями.

Функція *calloc()* повертає покажчик на перший байт виділеної області пам'яті. Якщо для задоволення запиту немає достатнього обсягу неперервної пам'яті, повертається нульовий покажчик. **Перед спробою використати виділену пам'ять обов'язково слід перевірити, що значення, яке повернула функція *calloc()*, не дорівнює нулю!**

Для зміни розміру блоку раніше виділеної пам'яті слід використовувати функцію *realloc()*. Її прототип наступний:

```
void* realloc(void* ptr, size_t size);
```

Дія функції *realloc()* у версії C99 трохи відрізняється від її роботи в C89, хоча кінцевий результат однаковий. У C89 функція *realloc()* змінює розмір блоку раніше виділеної пам'яті, що адресується покажчиком *ptr* відповідно до заданого розміру *size*. Значення параметра *size* може бути більшим або меншим, ніж вихідний розмір блоку пам'яті. Функція *realloc()* повертає покажчик на блок пам'яті, оскільки не виключена необхідність переміщення цього блоку (наприклад при збільшенні розміру блоку пам'яті). У цьому випадку вміст старого блоку (до *size* байтів) копіюється в новий блок.

У версії C99 блок пам'яті, що адресується параметром *ptr*, звільняється, а замість нього виділяється новий блок. Вміст нового блоку збігається з вмістом вихідного (принаймні збігаються перші *size* байтів). Функція повертає покажчик

на новий блок. Причому допускається, щоб новий і старий блоки починалися з однакової адреси (тобто покажчик, який повертається функцією *realloc()*, може збігатися з покажчиком, переданим в параметрі *ptr*).

Якщо значення покажчика *ptr* при виклику *realloc()* дорівнює нулю, функція *realloc()* просто виділяє *size* байтів пам'яті і повертає покажчик на цю пам'ять. Тобто, у цьому випадку вона працює як функція *malloc()*.

Якщо значення параметра *size* при виклику функції дорівнює нулю, пам'ять, що адресується параметром *ptr*, звільняється. Тобто, у цьому випадку вона працює як функція *free()*.

Якщо в динамічно розподіленій області пам'яті немає достатнього обсягу вільної пам'яті для виділення *size* байтів, повертається нульовий покажчик, а вихідний блок пам'яті залишається незмінним.

Ще один спосіб звільнення раніше виділеної пам'яті пов'язаний з використанням функції *free()*. Дія функції *free()* протилежна дії функцій *malloc()* та *calloc()* в тому сенсі, що функція повертає системі блок пам'яті, виділений раніше за допомогою функцій виділення пам'яті. Іншими словами, вона звільняє ділянку пам'яті, яка може бути знову використаний функцією *malloc()*. Функція *free()* має наступний прототип:

```
void free (void* p);
```

Тут *p* - покажчик на ділянку пам'яті, виділена перед цим функцією *malloc()*. Функцію *free()* ні в якому разі не можна викликати з неправильним аргументом (тобто, з покажчиком, на який раніше не було розміщено ніякого блоку даних), - це миттєво зруйнує всю систему розподілу пам'яті і призведе до аварійної зупинки програми.

Лістинг 5.4. Приклад роботи з елементами двовимірного динамічного масиву за допомогою покажчиків і роботи з ним за допомогою адресної арифметики:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main (void) {
    int i, /* лічильник рядків */
        j, /* лічильник стовпців */
        n, /* поточне число рядків у масиві */
        m, /* поточне число стовпців в масиві */
        **pi = NULL; /* Покажчик на сам двовимірний динамічний масив */

    printf ("Введіть число рядків і стовпців:");
    scanf ("%d %d", &n, &m);

    /* Виділення пам'яті під масив з одночасним контролем наявності вільної
    пам'яті */
    if (! (pi = (int**) malloc (n*m*sizeof (int)))) {
        printf ("Для такого масиву не вистачає пам'яті!");
        getch (); /* Затримка! */
        return 1;
    }
```

```

}

/* Заповнення масиву числами */
for (i = 0; i < n; ++ i)
    for (j = 0; j < m; ++j)
        *((int*) pi + i*m + j) = (i*m) + ji + 1; /* +1, Тому, що індексація з нуля! */

/* Друк масиву */
for (i = 0; i < n; ++ i) {
    for (j = 0; j < m; ++ j) /* друк поточного i-го рядка масиву */
        printf ("%3d», *((int *) pi + i*m + j));
    printf ("\n"); /* Перехід до друку нового рядка */
}

getch (); /* Затримка! */
free (pi);
return 0;
}

```

Лістинг 5.5. Приклад реалізації двовимірного динамічного масиву за допомогою покажчиків і роботи з ним за допомогою індексів:

```

int main(int argc, CHAR* argv[]) {
    int i, /* лічильник рядків */
        j, /* лічильник стовпчиків*/
        n, /* поточне число рядків в масиві */
        m, /* поточне число стовпчиків в масиві */
        **pi = NULL; /* вказівник на сам двовимірний динамічний масив */

    setlocale (LC_ALL, "RUS");
    printf ("Введіть число рядків і стовпчиків:");
    scanf ("%d %d", &n, &m);

    /* Виділення пам'яті під масив з одночасним контролем наявності вільної
    пам'яті */
    pi = (int**) malloc (n*sizeof(int*));
    if ( !pi ) {
        printf("Для такого масива не має вільної пам'яті!");
        getch(); /* затримка! */
        return 1;
    }

    for (i = 0; i < n; i++) {
        pi[i] = (int*) malloc(m * sizeof(int));
        if ( !pi[i] ) {
            printf("Для такого масива не має вільної пам'яті!");
            getch(); /* затримка! */
            return 1;
        }
    }
}

```

```

/* Заповнення масиву числами */
for (i = 0; i < n; ++i)
    for (j = 0; j < m; ++j)
        pi[i][j] = (i*m)+j+1;          /* +1, так як індексація з нуля! */

/* Друк масиву */
for (i = 0; i < n; ++i) {
    for (j = 0; j < m; ++j)          /* друк поточного i-го рядка масиву */
        printf ("%3d", pi[i][j]);
    printf ("\n");                  /* перехід до друку нового рядка */
}

getch();    /* затримка! */

for (i = 0; i < n; i++)              /* звільнення пам'яті під рядки масиву */
    free (pi[i]);
free (pi);                          /* звільнення пам'яті під масив покажчиків на рядки
                                     масиву */

return 0;
}

```

5.2. Варіанти завдань

Увага! Всі завдання виконуються двома способами, з використанням як арифметики покажчиків (Лістинг 5.4), так і роботи за допомогою індексів (Лістинг 5.5), виключно з використанням покажчиків для індексації та роботи з елементами масивів. Розміри масиву повинні задаватися при початку роботи програми з клавіатури. Після виконання окремої операції завдання результат операції повинен виводитися на екран

Варіант	Завдання
1	На основі початкових масивів A і B з робочим розміром n сформувати масив C, елементи якого обчислюються за формулою: $C_i = A_i * B_i$. Упорядкувати масив C по зростанню, використовуючи метод вибору. Вивести елементи масиву C на екран. Масиви A, B і C є цілочисельними. Значення n і значення елементів масиву A і B вводяться з клавіатури. При цьому значення елементів всередині масиву A не повинні повторюватися.
2	На основі початкових масивів A і B з робочим розміром n сформувати масив C, елементи якого обчислюються за формулою: $C_i = A_i + B_i$. Упорядкувати масив C по зростанню, використовуючи метод «бульбашки». Вивести масиви A, B і C на екран. Масиви A, B і C складаються з дійсних чисел. Значення n вводиться користувачем з клавіатури. Значення елементів масивів A і B генеруються за допомогою датчика випадкових чисел. При цьому значення елементів всередині масиву B не повинні повторюватися.
3	На основі початкових масивів A[n] і B[m] (n і m - робочі розміри масивів) сформувати масив C, який складатиметься з чисел, які одночасно входять як в масив A, так і в масив B. Упорядкувати масив C по зростанню, використовуючи метод вибору. Вивести елементи масиву C на екран. Масиви A, B і C є цілочисельними. Значення m і n, а також значення елементів масиву A і B вводяться з клавіатури

Варіант	Завдання
4	На основі початкових масивів $A[n]$ і $B[m]$ (n і m - робочі розміри масивів) сформувати масив C, який складатиметься з чисел, які входять в масив A, але при цьому не входять в масив B. Впорядкувати масив C по зростанню, використовуючи метод «бульбашки». Вивести елементи масиву C на екран. Масиви A, B і C є цілочисельними. Значення m і n, а також значення елементів масиву A і B вводяться з клавіатури.
5	На основі початкових масивів $A[n]$ і $B[m]$ (n і m - робочі розміри масивів) сформувати масив C, який складатиметься з чисел, які входять в масив B, але при цьому не входять в масив A. Впорядкувати масив C по зростанню, використовуючи метод «бульбашки». Вивести елементи масивів A, B і C на екран. Масиви A, B і C є цілочисельними. Значення m і n вводяться користувачем з клавіатури. Значення елементів масивів A і B генеруються за допомогою датчика випадкових чисел.
6	На основі початкових масивів $A[n]$ і $B[m]$ (n і m - робочі розміри масивів) сформувати двовимірний масив $C[n][m]$, елементи якого будуть обчислюватися по формулі $C_i^j = A_i * B_j$. Вивести елементи масивів A, B і C на екран. Масиви A, B і C є цілочисельними. Значення m і n, а також значення елементів масивів A і B вводяться користувачем з клавіатури. Перед заповненням масиву C, слід упорядкувати масиви A і B за зростанням, використовуючи метод вибору.
7	На основі початкових масивів $A[n]$ і $B[m]$ (n і m - робочі розміри масивів) сформувати двовимірний масив $C[n][m]$, елементи якого будуть обчислюватися по формулі $C_i^j = A_i * B_j$. Вивести елементи масивів A, B і C на екран. Масиви A, B і C містять дійсні числа. Значення m і n вводяться користувачем з клавіатури. Значення елементів масивів A і B генеруються за допомогою датчика випадкових чисел. Перед формуванням масиву C, слід упорядкувати масиви A і B за зростанням, використовуючи метод «бульбашки».
8	На основі початкових масивів A і B з робочим розміром n сформувати двовимірний масив $C[n][n]$, елементи якого будуть обчислюватися по формулі: $C_j = A_i^2 - B_j^2$. Вивести елементи масивів A, B і C на екран. Масиви A, B і C містять цілі числа. Значення n вводяться користувачем з клавіатури. Значення елементів масивів A і B генеруються за допомогою датчика випадкових чисел.
9	На основі початкового двовимірного масиву C з робочим розміром $n \times n$ сформувати масив A з розміром n, i-й елемент якого буде дорівнювати сумі елементів i-го рядка масиву C $A_i = \sum_{j=0}^{n-1} C_i^j.$ Упорядкувати масив A за спаданням, використовуючи метод «бульбашки». Вивести масиви A і C на екран. Масиви A і C повинні містити цілі числа. Значення n вводяться користувачем з клавіатури. Значення елементів масиву C генеруються за допомогою датчика випадкових чисел.

Варіант	Завдання
10	На основі початкового двовимірного масиву С з робочим розміром $n \times m$ сформувати масив А з розміром n, i-й елемент якого буде дорівнювати добутку елементів i-го рядка масиву С: $A_i = \prod_{j=0}^{m-1} C_i^j.$ Упорядкувати масив А за спаданням, використовуючи метод вибору. Вивести масиви А і С на екран. Масиви А і С повинні містити дійсні числа. Значення n і m вводяться користувачем з клавіатури. Значення елементів масиву С також вводяться користувачем з клавіатури.
11	На основі початкового двовимірного масиву С з робочим розміром $n \times m$ сформувати масив В з розміром n, i-й елемент якого буде дорівнювати середньому арифметичному значенню для елементів i-го рядка масиву С: $B_i = \left(\sum_{j=0}^{m-1} C_i^j \right) / m.$ Упорядкувати масив В по зростанню, використовуючи метод «бульбашки». Вивести масиви В і С на екран. Масив С повинен складатися з цілих чисел, масив В - з дійсних. Значення n і m, а також значення елементів масиву С вводяться користувачем з клавіатури.
12	Використовуючи метод «бульбашки», впорядкувати по зростанню значення, що містяться в рядках двовимірного масиву С з робочим розміром $n \times m$. Вивести масив С на екран. Масив С повинен складатися з цілих чисел. Значення n і m, а також значення елементів масиву С вводяться користувачем з клавіатури. Елементи в кожному рядку масиву С не повинні повторюватися.
13	Використовуючи метод вибору, впорядкувати за спаданням значення, що містяться в рядках двовимірного масиву С з робочим розміром $n \times n$. Вивести масив С на екран. Масив С повинен складатися з цілих чисел. Значення n вводиться користувачем з клавіатури. Значення елементів масиву С генеруються за допомогою датчика випадкових чисел. Елементи в рядках масиву С не повинні повторюватися.
14	На основі початкового двовимірного масиву С з робочим розміром $n \times n$, сформувати одновимірний масив А. Масив А повинен містити тільки ті числа, які зустрічаються в масиві С тільки один раз. Упорядкувати масив А за спаданням, використовуючи метод вибору. Вивести масив А на екран. Масиви А і С повинні складатися з цілих чисел. Значення n, а також значення елементів масиву С вводяться користувачем з клавіатури.

Варіант	Завдання
15	На основі початкового двовимірного масиву С з робочим розміром $n \times n$, сформуванати одновимірний масив А. Масив А повинен містити тільки ті числа, які зустрічаються в масиві С більше одного разу. Упорядкувати масив А за спаданням, використовуючи метод «бульбашки». Вивести масиви С і А на екран. Масиви А і С повинні складатися з цілих чисел. Значення n вводиться користувачем з клавіатури. Значення елементів масиву С генеруються за допомогою датчика випадкових чисел.
16	Написати і відлагодити програму, в якій: – з клавіатури вводиться двовимірний масив дійсних чисел розміром $n \times n$; – перевіряється правильність прийому масиву; – рядки масиву сортуються за збільшенням; – елементи 1-го рядка і 1-го стовпця міняються місцями. Результати всіх дій виводити на екран.
17	Написати і відлагодити програму, в якій: – з клавіатури вводиться двовимірний масив дійсних чисел розміром $n \times n$; – перевіряється правильність прийому масиву; – рядки масиву сортуються за збільшенням; – елементи 1-го рядка замінюються нулями, останнього - одиницями. Результати всіх дій виводити на екран.
18	Написати і відлагодити програму, в якій: – з клавіатури вводиться двовимірний масив дійсних чисел розміром $n \times n$; – перевіряється правильність прийому масиву; – рядки масиву сортуються за збільшенням; – елементи головної діагоналі масиву виводяться на екран. Результати всіх дій виводити на екран.
19	Написати і відлагодити програму, в якій: – з клавіатури вводиться двовимірний масив дійсних чисел розміром $n \times n$; – перевіряється правильність прийому масиву; – рядки масиву сортуються за збільшенням; – елементи 1-ого і останнього рядка міняються місцями. Результати всіх дій виводити на екран.
20	Написати і відлагодити програму, в якій: – з клавіатури вводиться двовимірний масив дійсних чисел розміром $n \times n$; – перевіряється правильність прийому масиву; – рядки масиву сортуються по убутанню. – визначається середнє арифметичне всіх елементів масиву. Результати всіх дій виводити на екран.

Варіант	Завдання
21	Написати і відлагодити програму, в якій: – з клавіатури вводиться двовимірний масив дійсних додатних чисел розміром $n \times n$; – перевіряється правильність прийому масиву; – здійснюється симетрична заміна елементів головної та бічної діагоналі початкового масиву; – рядки масиву сортуються за спаданням. Результати всіх дій виводити на екран.
22	На основі початкового двовимірного масиву C з робочим розміром $n \times n$, сформувати два одновимірні масиви A і B такі, що в кожному з них міститься по половині елементів початкового масиву C (елементи для вставки обирати по збільшенню значень індексів: наприклад, при розмірності масиву C 4×4, в масив A записати 8 елементів – перші два рядки C, а в масив B – теж 8 елементів – другі два рядки C). Масив A впорядкувати за спаданням, а масив B – за зростанням, використовуючи метод «бульбашки». Вивести масиви C, A і B на екран. Масиви A, B, C повинні складатися з цілих чисел. Значення n та елементи масиву C генерується генератором випадкових чисел і завжди є парним.
23	На основі початкових масивів A і B з робочим розміром n сформувати двовимірний масив $C[n][n]$, кожен елемент $C_{i,j}$ якого буде обчислюватися, як $C_{ij} = A_i + B_j$. Вивести елементи масивів A, B і C на екран. Масиви A, B і C містять цілі числа. Значення n вводяться користувачем з клавіатури. Значення елементів масивів A і B генеруються за допомогою генератора випадкових чисел.
24	На основі початкових масивів $A[n]$ і $B[m]$ (n і m - робочі розміри масивів) сформувати масив C, який складатиметься з чисел, які не входять в масив A, але входять в масив B. Впорядкувати масив C по спаданню, використовуючи метод вибору. Вивести елементи масиву C, A і B на екран. Масиви A, B і C є цілочисельними. Значення m і n, а також значення елементів масиву A і B вводяться з клавіатури.
25	На основі початкових масивів A і B з робочим розміром n сформувати масив C розміром n, елементи якого обчислюються як: $C_i = A_i - B_i$. Упорядкувати масив C спочатку по зростанню, потім – за спаданням, використовуючи метод вибору. Вивести елементи масиву C на екран до і після обох видів сортувань. Масиви A, B і C є цілочисельними. Значення n і значення елементів масиву A і B вводяться з клавіатури. При цьому значення елементів всередині масиву B не повинні повторюватися.

Варіант	Завдання
26	На основі початкових масивів $A[n]$ і $B[m]$ (n і m - робочі розміри масивів) сформувати двовимірний масив $C[n][m]$, елементи якого будуть обчислюватися по формулі $C_i^j = A_i + B_j$. Вивести елементи масивів A, B і C на екран. Масиви A, B і C містять дійсні числа. Значення m і n вводяться користувачем з клавіатури. Значення елементів масиву A генеруються за допомогою генератора випадкових чисел. Значення масиву B – є значення масиву A, але в оберненому порядку ($B_1 = A_n, B_2 = A_{n-1}, \dots, B_m = A_1$). Перед формуванням масиву C, слід упорядкувати масиви A і B за зростанням, використовуючи метод «бульбашки».
27	На основі початкового двовимірного масиву C з робочим розміром $n \times n$, сформувати одновимірний масив A. Масив A повинен містити тільки ті числа, які зустрічаються в масиві C тільки один раз і є додатними. Упорядкувати масив A за спаданням, використовуючи метод «бульбашки». Вивести масиви C і A на екран. Масиви A і C повинні складатися з цілих чисел. Значення n вводиться користувачем з клавіатури. Значення елементів масиву C генеруються за допомогою генератора випадкових чисел в діапазоні від 0 до 100.
28	Написати і відлагодити програму, в якій: – з клавіатури вводиться двовимірний масив дійсних чисел розміром $n \times n$; – перевіряється правильність прийому масиву; – рядки масиву сортуються за збільшенням; – елементи бічної діагоналі замінюються нулями, головної - одиницями. – Підраховується сума елементів масиву за модулем. Результати всіх дій виводити на екран.
29	Написати і відлагодити програму, в якій: – з клавіатури вводиться двовимірний масив дійсних чисел розміром $n \times m$; – перевіряється правильність прийому масиву; – рядки масиву сортуються за спаданням; – визначається сума елементів кожного рядка; – визначається сума елементів кожного стовбця; Результати всіх дій виводити на екран.
30	Написати і відлагодити програму, в якій: – користувач вводить ціле число n – розмірність масиву; – за допомогою генератора випадкових чисел генерується двовимірний масив цілих дійсних додатних чисел розміром $n \times n$ в діапазоні чисел від 1 до 20; – рядки масиву сортуються за спаданням; – елементи n-го рядка і n-го стовпця міняються місцями. Результати всіх дій виводити на екран.

5.3. Зміст звіту

1. Титульний лист з назвою комп'ютерного практикуму, номером

- варіанту, прізвищем студента та групи.
2. Мета комп'ютерного практикуму.
 3. Короткі теоретичні відомості.
 4. Завдання.
 5. Схема алгоритму.
 6. Текст програми.
 7. Результати роботи програми.
 8. Висновки про отримані результати роботи програми.

5.4. Контрольні запитання

1. Яке загальне призначення вказівників в мові C?
2. Які арифметичні операції допускаються для вказівників?
3. Які унарні оператори використовуються з вказівниками? Як вони називаються?
4. Для яких типів даних може бути використаний вказівник?
5. Як числові значення вказівників змінюються при їх інкрементуванні в залежності від типів даних?
6. За допомогою якого формату здійснюється виведення на консоль адрес змінних заданого типу?
7. Що таке багаторівнева адресація? Як вона організовується в мові C?
8. Як здійснюється ініціалізація вказівників на речові типи даних?
9. Як здійснюється ініціалізація вказівників на символьний тип даних?
10. Який сенс має значення вказівника NULL?
11. Що відбудеться, якщо застосувати до вказівника зі значенням NULL операцію розіменування?
12. Як слід визначати і ініціалізувати вказівника константу?
13. Як слід визначати і ініціалізувати константні вказівники?
14. Яка відмінність константного вказівника від вказівника на константу?

Комп'ютерний практикум № 6. ФУНКЦІЇ

Мета комп'ютерного практикуму: навчитися розробляти і програмувати власні функції і вивчити правила роботи з ними.

6.1. Теоретичні відомості

У раніше розглянутих комп'ютерних практикумах неодноразово використовувалися різні функції бібліотек, які підключаються. Разом з тим існуючих функцій мови C часто недостатньо для написання власних програм і виникає необхідність створення своїх власних функцій. У зв'язку з цим потрібно розуміти, в яких випадках доцільно створювати свої функції. Зазвичай це робиться для позбавлення програміста від багаторазового писання одного і того самого коду в програмі. Наприклад, якщо часто виконуються дії копіювання одного рядка в інший, то таку операцію краще визначити у вигляді функції і використовувати її в міру необхідності. Для оголошення функції використовується наступний синтаксис:

<тип> <ім'я функції> ([список параметрів]) {<тіло функції>}

тип визначає тип значення, що повертається при закінченні виконання функції. *ім'я_функції* служить для її виклику в програмі. Правила вибору імені функції збігаються з правилом визначення імен змінних. *список_параметрів* необхідний для передачі функції при її виклику яких-небудь даних, необхідних для її роботи. *тіло_функції* - це послідовність (блок) операторів, які виконуються при її виклику. Наступний приклад ілюструє створення функції користувача для зведення числа в квадрат.

Лістинг 6.1. Приклад задання функції.

```
double square (double x) {  
    x = x*x;  
    return x;  
}  
  
int main () {  
    double arg = 5;  
    double sq1 = square (arg);  
    double sq2 = square (3);  
    return 0;  
}
```

У даному прикладі задається функція з ім'ям *square*, яка приймає один вхідний параметр типу *double*, підносить його до квадрату і повертає обчислене значення програмі, яка її викликає за допомогою оператора *return*. Слід зазначити, що робота функції завершується при виконанні оператора *return*. Навіть якщо після цього оператора будуть знаходитись інші оператори, то вони виконуватися вже не будуть. Наприклад,

```
int square (int x) {  
    x = x*x;  
    return x;  
}
```

```
printf ("%d", x);
}
```

При виклику даної функції оператор *printf()* не буде виконаний ніколи, тому що оператор *return* завершить ще до нього роботу функції *square()*. Оператор *return* є обов'язковим, якщо функція повертає які-небудь значення. Якщо ж вона має тип *void*, тобто нічого не повертає, то оператор *return* може не використовуватися.

Користуючись розглянутими правилами, можна створювати безліч своїх функцій. При цьому важливо, щоб оголошення функції при компіляції програми зустрілося компілятору раніше, ніж її використання в програмі, подібно змінним. Саме тому у всіх прикладах вище оголошення функцій здійснювалось до функції *main()*, в якій вони викликаються.

Функція може приймати довільне число аргументів, але повертає тільки один або жодного (тип *void*). Для завдання кількох аргументів функції використовується наступна конструкція:

```
void show (int x, int y, int z) {}
```

Тут слід звернути увагу на те, що кожній змінній в списку аргументів функції передують її тип. На відміну від оголошення звичайних змінних. Тому наступний програмний рядок приведе до повідомлення про помилку на етапі компіляції:

```
void show (int x, y, z) {}
```

// невірне оголошення

Якщо кількість функцій користувача велика (50 і вище), то виникає незручність в їх візуальному представленні в загальному тексті програми. Дійсно, маючи список з 100 різних функцій з їх реалізаціями, в них стає складно орієнтуватися і вносити необхідні зміни. Для вирішення даної проблеми в мові C при створенні своїх функцій можна користуватися правилом: спочатку задаються оголошення функції (прототипи функції), а потім їх реалізації.

Мова C дозволяє задавати функції з однаковими іменами, але різними типами вхідних аргументів. Наступний приклад демонструє зручність використання таких функцій при їх виклику.

Лістинг 6.2. Приклад використання перевантажених функцій.

```
#include <stdio.h>
```

```
double abs(double arg); /* три прототипи функцій */
float abs(float arg);
int abs(int arg);
```

```
int main () {
    double a_d = -5.6;
    float a_f = -3.2;
    int a_i;
    a_d = abs (a_d);
    a_f = abs (a_f);
    a_i = abs (-8);
}
```

```

    return 0;
}

double abs(double arg) { /* реалізації усіх трьох функцій */
    if (arg < 0)
        arg = arg * (-1);
    return arg;
}

float abs(float arg) {
    return (arg < 0)? -arg: arg;
}

int abs(int arg) {
    return (arg < 0)? -arg: arg;
}

```

У представленій програмі задаються три функції з ім'ям *abs* і різними вхідними та вихідними аргументами для обчислення модуля числа. Завдяки такому оголошенню при обчисленні модуля різних типів змінних у функції *main()* використовується виклик функції з одним і тим самим ім'ям *abs*. При цьому компілятор в залежності від типу змінної автоматично вибере потрібну функцію. Такий підхід до оголошення функцій називається перевантаженням.

У мові C можна задавати значення аргументів функції, які будуть використовуватися за замовчуванням, тобто якщо програміст не введе своє значення. Наведений нижче фрагмент програми демонструє правило використання аргументів за замовчуванням:

```

void some_func(int a = 1, int b = 2, int c = 3) {
    printf ("a = %d, b = %d, c = %d \n", a, b, c);
}

```

Завдяки початковій ініціалізації значень змінних, функція *some_func()* може бути викликана з різним набором аргументів:

```

int main(void) {
    show_func();
    show_func(10);
    show_func(10,20);
    show_func(10,20,30);
    return 0;
}

```

В результаті, на екрані з'являться наступні рядки:

```

a = 1, b = 2, c = 3
a = 10, b = 2, c = 3
a = 10, b = 20, c = 3
a = 10, b = 20, c = 30

```

З отриманого результату видно, що за замовчуванням значення аргументів рівні встановленим значенням при визначенні функції. У разі

введення нових значень, змінні *a*, *b* і *c* відповідно змінюють свої значення на введені.

При використанні значень аргументів за замовчуванням слід користуватися правилом: аргументи зі значеннями за замовчуванням повинні знаходитись в списку аргументів функції останніми. Наступний приклад показує правильні і неправильні оголошення функцій:

```
void my_func(int a, int b = 1, int c = 1); /* Правильне оголошення void
                                           my_func (int a, int b, int c = 1) */
void my_func(int a = 1, int b, int c = 1); /* Неправильне оголошення void
                                           my_func (int a, int b = 1, int c) */
```

У мові С допускається щоб функція викликала саму себе. Цей процес називається *рекурсією*. У деяких завданнях з програмування такий підхід дозволяє помітно спростити створюваний програмний код. Розглянемо даний процес на наступному прикладі.

Лістинг 6.3. Приклад використання рекурсивних функцій.

```
#include <stdio.h>

void up_and_down(int);

int main(void) {
    up_and_down(1);
    return 0;
}

void up_and_down(int n) {
    printf ("Рівень вниз %d\n", n);
    if (n < 4)
        up_and_down (n + 1);
    printf ("Рівень вгору %d\n", n);
}
```

Результатом роботи цієї програми буде виведення на екран наступних рядків:

```
Рівень вниз 1
Рівень вниз 2
Рівень вниз 3
Рівень вниз 4
Рівень вгору 4
Рівень вгору 3
Рівень вгору 2
Рівень вгору 1
```

Отриманий результат роботи програми пояснюється наступним чином. Спочатку функція *main()* викликає функцію *up_and_down()* з аргументом 1. У результаті аргумент *n* даної функції приймає значення 1 і функція *printf()* друкує перший рядок. Потім виконується перевірка і якщо *n < 4*, то знову викликається функція *up_and_down()* з аргументом на 1 більше (*n+1*). В

результаті знову викликається та сама функція, яка друкує другий рядок. Даний процес продовжується до тих пір, поки значення аргументу не стане рівним 4. У цьому випадку оператор *if* не передасть управління на *up_and_down()* і, нарешті, буде викликана функція *printf()*, яка друкує п'ятий рядок «Рівень вгору 4». Потім функція завершує свою роботу і управління передається функції, яка викликала дану функцію. Це функція *up_and_down()* з аргументом $n=3$, яка також продовжує свою роботу і переходить до оператора *printf()*, який друкує 6 рядок «Рівень вгору 3». Цей процес продовжується до тих пір, поки не буде досягнутий вихідний рівень, тобто перший виклик функції *up_and_down()* і керування знову буде передано функції *main()*, яка завершить роботу програми.

6.2. Завдання на комп'ютерний практикум

1. Написати дві програми по роботі з функціями у відповідності з номером свого варіанту.

6.3. Варіанти завдань

Варіант	1-е завдання (без рекурсії)	2-е завдання (з рекурсією)
1	Написати функцію обчислення площі прямокутника	За допомогою рекурсивної функції здійснити обчислення ряду Фібоначчі з початковими членами a_0 та a_1 . В задачі не можна використовувати цикли та
2	Написати функцію обчислення периметра прямокутника	За допомогою рекурсивної функції здійснити виведення на екран елементів одновимірного масиву розміру N . В задачі не можна використовувати цикли
3	Написати функцію обчислення довжини кола	Дано два цілих числа A і B . За допомогою рекурсивної функції вивести всі числа від A до B включно, в порядку зростання, якщо $A < B$, або в порядку спадання в іншому випадку. У задачі не можна використовувати цикли і масиви.
4	Написати функцію обчислення площі круга	За допомогою рекурсивної функції здійснити пошук максимального елемента одновимірного масиву розміру N . У задачі не можна використовувати цикли

Варіант	1-е завдання (без рекурсії)	2-е завдання (з рекурсією)
5	Написати функцію обчислення об'єму паралелепіпеда	За допомогою рекурсії обчислити значення функції Аккермана $A(m, n)$, що визначається таким чином: $A(m, n) = \begin{cases} n + 1, & \text{при } m = 0 \\ A(m - 1, 1) & \text{при } n = 0 \\ A(m - 1, A(m, n - 1)) & \text{при } m > 0, n > 0 \end{cases}$ для чисел m и n. У задачі не можна використовувати цикли.
6	Написати функцію обчислення евклідової відстані між двома точками у просторі, що задаються своїми декартовими координатами	За допомогою рекурсивної функції здійснити пошук мінімального елемента одновимірного масиву. У задачі не можна використовувати цикли
7	Написати функцію обчислення суми елементів масиву розмірністю N	Дано натуральне число N . Використовуючи рекурсію вивести слово YES, якщо число N є точним ступенем двійки, або слово NO в іншому випадку. Операцією піднесення до степеня користуватися не можна! Масиви і цикли використовувати не можна
8	Написати функцію знаходження максимального значення елемента масиву розмірністю N	За допомогою рекурсивної функції обчислити суму елементів одновимірного масиву розміру N . В задачі цикли використовувати не можна
9	Написати функцію знаходження мінімального значення елемента масиву розмірністю N	Дано натуральне число N . За допомогою рекурсивної функції обчислити суму його цифр. При виконанні цього завдання не можна використовувати рядки, списки, масиви і цикли.

Варіант	1-е завдання (без рекурсії)	2-е завдання (з рекурсією)
10	Написати функцію обчислення добутку елементів масиву розмірністю N	За допомогою рекурсивної функції обчислити середнє арифметичне елементів одновимірного масиву розміру N . В задачі не можна використовувати цикли
11	Написати не рекурсивну функцію обчислення факторіала введеного з клавіатури числа N	Дано натуральне число N . Виведіть всі його цифри по одній, в звичайному порядку, розділяючи їх пробілами або новими рядками. В задачі не можна використовувати цикли
12	Написати функцію перетворення номера місяця в його назву	Дано слово, що складається лише з малих латинських букв. За допомогою рекурсивної функції перевірте, чи є це слово паліндромом. Виведіть YES або NO.
13	Написати функцію, яка повертає 1, якщо отриманий символ ch , що є аргументом, є голосною літерою української абетки, і нуль - в іншому випадку	За допомогою рекурсивної функції обчислити факторіал числа N , введенного з клавіатури. У задачі не можна використовувати цикли
14	Написати функцію, яка повертає перетворений до верхнього регістру рядок символів, отриманий в якості аргументу	Написати рекурсивну функцію переведення натурального числа Z у систему числення з основою N ($N > 1$)
15	Написати функцію, яка обчислює опір ланцюга, що складається з двох резисторів. Параметрами функції повинні бути величини опорів резисторів, а також тип з'єднання (послідовне або паралельне)	Написати рекурсивну функцію обчислення степеня дійсного числа з цілим показником x^n , згідно з формулою: $x^n = \begin{cases} 1, & n = 0 \\ \frac{1}{x^{ n }}, & n < 0 \\ x * x^{n-1}, & n > 0 \end{cases}$

Варіант	1-е завдання (без рекурсії)	2-е завдання (з рекурсією)
16	Написати функцію, що забезпечує розв'язання квадратного рівняння. Параметрами функції повинні бути коефіцієнти рівняння. Повертати функція повинна кількість неоднакових коренів та самі корені	Написати рекурсивну функцію множення двох цілих чисел. Стандартною операцією множення (*) користуватися заборонено
17	Написати функцію переведення градусів Цельсія у градуси Фаренгейта	За допомогою рекурсії реалізувати операцію ділення одного цілого числа на інше. Функція повинна повертати результат та залишок від ділення. Стандартними операціями ділення (/ і %) користуватися заборонено
18	Написати функцію обчислення вартості 1 км їзди на автомобілі. Параметрами функції повинні бути витрата автомобіля (л/100 км) і вартість 1 л палива	Написати рекурсивну функцію обчислення суми цифр натурального числа N
19	Написати функцію перекладу 100 бальної шкали оцінювання в шкалу ECTS	За допомогою рекурсії обчислити n-й член арифметичної прогресії
20	Написати функцію, яка порівнює два цілих числа і повертає результат порівняння у вигляді одного із символів: '>', '<' або '='	Написати рекурсивну функцію обчислення НСД (A, B) цілого типу, яка знаходить найбільший спільний дільник (НСД) двох натуральних чисел A і B, використовуючи алгоритм Евкліда: $\text{НСД}(A, B) = \text{НСД}(B, A \% B)$, якщо $B \neq 0$; $\text{НСД}(A, 0) = A$
21	Вивести на екран перші n рядків трикутника Паскаля. Реалізувати нерекурсивний варіант функції	Написати рекурсивну функцію, що для даного дійсного аргументу перевіряла б, чи є він натуральним числом. При цьому обов'язково використовуйте рекурсивне визначення натурального числа.
22	Написати функцію обчислення площі кола та застосувати її для обчислення площі кільця (як різницевої площі)	Створити рекурсивну функцію для обрахунку значення виразу $\sum_{i=1}^n (1 + \sin i)$, де n-ціле натуральне число

Варіант	1-е завдання (без рекурсії)	2-е завдання (з рекурсією)
23	Дано парне число $n > 2$. Перевірити для нього гіпотезу Гольдбаха: кожне парне число $n > 2$ можна зобразити у вигляді суми двох простих чисел. Написати функцію, яка перевіряє, чи є число простим	Реалізувати рекурсивну функцію <i>Incl_calc()</i> , яка визначає кількість входжень заданого символу <i>ch</i> в рядок <i>st</i> . Визначитись з оптимальною кількістю параметрів функції
24	Написати функцію, яка обчислює суму елементів головної діагоналі масиву <i>A</i> дійсних чисел, розмірності $N \times N$	Реалізувати рекурсивну функцію <i>Count()</i> , яка повертає кількість елементів одновимірного масиву розміру <i>N</i> натуральних чисел, більших заданого числа <i>k</i> . Визначитись з оптимальною кількістю параметрів функції
25	<i>Щасливим</i> називають таке шестизначне натуральне число, у якого сума перших трьох цифр дорівнює сумі останніх трьох цифр. Знайти всі щасливі числа. Написати функцію для визначення суми цифр тризначного цілого числа	Вивести на екран перші <i>n</i> рядків трикутника Паскаля. Реалізувати рекурсивний варіант функції
26	Написати функцію, яка реалізує заміну елементів головної діагоналі масиву <i>A</i> , розмірності $N \times N$ на нулі	Реалізувати рекурсивну функцію, яка реалізує підрахунок кількості елементів масиву <i>A[n][n]</i> менших заданого числа <i>m</i>
27	Написати функцію, яка визначає індекс маси тіла за формулою: $I = \frac{m}{h^2}$, де <i>m</i> -маса тіла, <i>h</i> - зріст.	Дано натуральне число <i>N</i> . За допомогою рекурсивної функції вивести всі його цифри по одній, у зворотному порядку, розділяючи їх крапками. В задачі не можна використовувати цикли
28	Дано натуральне число <i>n</i> і послідовність натуральних чисел $a_1, a_2, \dots, a_n$. Показати всі елементи послідовності, які є повними квадратами. Написати функцію визначення повного квадрату числа	Написати рекурсивну функцію, яка обчислює суму $\sum_{i=1}^n x^{\frac{1}{i}},$ де <i>i</i> - ціле натуральне число

Варіант	1-е завдання (без рекурсії)	2-е завдання (з рекурсією)
29	Написати функцію, яка виводить слово, введене з клавіатури в зворотному порядку	Написати рекурсивну функцію визначення максимальної цифри в представлення цілого числа
30	Дано натуральне число n і послідовність натуральних чисел $a_1, a_2, \dots, a_n$. Показати всі елементи послідовності, які є простими числами. Написати функцію визначення простого числа	Дано натуральне число $n > 1$. Написати рекурсивну функцію, яка виводить всі прості множники даного числа по спаданню з врахуванням кратності.

6.4. Зміст звіту

1. Титульний лист з назвою комп'ютерного практикуму, номером варіанту, прізвищем студента та групи.
2. Мета комп'ютерного практикуму.
3. Короткі теоретичні відомості.
4. Завдання.
5. Схеми алгоритмів.
6. Текст програм.
7. Результати роботи програм.
8. Висновки про отримані результати роботи програм.

6.5. Контрольні запитання

1. Запишіть прототип функції, яка приймає два цілочисельних аргумента і повертає дійсне число.
2. Припустимо, дані три функції: `int abs (int x); float abs (float x); long abs (long x);` Яка з цих трьох функцій буде викликана в рядку `float a = abs(-6);`?
3. Запишіть функцію піднесення числа до квадрату.
4. Дайте поняття рекурсії.
5. У яких завданнях доцільно використовувати рекурсивні функції?
6. Наведіть функцію з трьома аргументами, один з яких заданий зі значенням за замовчуванням.

Комп'ютерний практикум № 7. СТРУКТУРИ ТА ФАЙЛИ

Мета комп'ютерного практикуму: навчитися працювати зі структурним типом даних, в тому числі і при обробці файлів.

7.1. Теоретичні відомості

Робота зі структурами

При розробці програм важливим є вибір ефективного способу представлення даних. У багатьох життєвих випадках недостатньо оголосити просту змінну або масив, - потрібна більш гнучка форма представлення даних. Таким елементом може бути структура, яка дозволяє включати в себе різні типи даних, а також інші структури. Наведемо приклад, в якому використання структури дозволяє ефективно представити дані. Таким прикладом буде інвентарний перелік книг, в якому для кожної книги необхідно вказувати її найменування, автора і рік видання. Причому кількість книг може бути різною, але будемо вважати, що не більше 100. Для зберігання інформації про одну книжку доцільно використовувати структуру, яка задається в мові C за допомогою ключового слова *struct*, за яким слідує її ім'я (тег) структури. Саме визначення структури, тобто те, що вона буде містити, записується у фігурних дужках *{}*. В нашому випадку структура буде мати наступний вигляд:

```
struct book {  
    char title [100];    // найменування книги  
    char author [100]; // автор  
    int year;           // рік видання  
};
```

Така конструкція задає свого роду шаблон представлення даних, але не сам об'єкт, яким можна було б оперувати подібно змінній або масиву. Для того щоб оголосити змінну структурного типу з ім'ям *book* використовується такий запис:

```
struct book lib; // оголошується змінна типу book
```

Після оголошення змінної *lib* є можливість працювати зі структурою як з єдиним об'єктом даних, який має три поля: *title*, *author* і *year*. Звернення до того чи іншого поля структури здійснюється через крапку: *lib.title*, *lib.author* і *lib.year*. Таким чином, для занесення в структурну змінну інформації можна використовувати наступний фрагмент програми:

```
printf ("Введіть найменування книги:");  
scanf ("%s", lib.title);  
printf ("Введіть автора книги:"); scanf ("%s", lib.author);  
printf ("Введіть рік видання книги:"); scanf ("%d", &lib.year);
```

Після цього у відповідні поля буде записана введена з клавіатури інформація, яка надалі буде зберігатися в єдиній структурній змінній *lib*. Однак за умовами завдання необхідно здійснювати запис не по одній, а по 100 книжок. У цьому випадку доцільно використовувати масив структур типу

book, який можна задати наступним чином:

```
struct book lib[100];
```

У цьому випадку програму введення та зберігання інформації по книгах можна записати в наступному вигляді:

Лістинг 7.1. Інвентарний перелік книг.

```
#include <stdio.h>

struct book {
    char title[100];    // найменування книги
    char author[100];   // автор
    int year;           // рік видання
};

int main () {
    int cnt_book = 0, ch;
    struct book lib[100];
    do {
        printf ("Введіть найменування книги:");
        scanf ("%s", lib[cnt_book].title);
        printf ("Введіть автора книги:");
        scanf ("%s", lib [cnt_book].author);
        printf ("Введіть рік видання книги:");
        scanf ("%d", &libyear);
        printf ("Натисніть q для завершення введення:");
        cnt_book ++;
    }
    while (scanf ("%d", ch) == 1 && cnt_book <100);
    return 0;
}
```

Цей приклад показує зручність зберігання інформації по книжкам, використовуючи структурний тип даних. Той самий алгоритм в загальному випадку можна реалізувати і без структури, але тоді довелося б використовувати два окремих двовимірних масива символів з назвами *Найменування* і *Автор* та один одновимірний масив для зберігання року видання книги. Незважаючи на те, що формально такий запис був би коректним з огляду на правила мови С, але набагато менш зручним при обробці. Графічно масив структур можна представити у вигляді таблиці, в якій роль стовпців грають поля, а роль рядків елементи масиву - структури.

Структури можна автоматично ініціалізувати при їх оголошенні подібно масивам, використовуючи наступний синтаксис:

```
struct book lib = {
    "Євгеній Онєгін",
    "Пушкін О.С.", 1995
};
```

При виконанні даного фрагмента програми в поля структури *title*, *author* і *year* буде записана інформація: "Євгеній Онегін", "Пушкін О.С.", 1995. Тут слід звернути увагу, що послідовність даних при ініціалізації повинна відповідати послідовності полів у структурі. Це накладає певні обмеження, тому при ініціалізації необхідно пам'ятати послідовність полів у структурі. Стандарт C99 допускає більш гнучкий механізм ініціалізації полів структури, в якому можна недотримуватися жорсткої послідовності полів, як в опису типу структури:

```
struct book lib = {  
    .year = 1995,  
    .author = "Пушкін О.С.",  
    .title = "Євгеній Онегін"  
};
```

або

```
struct book lib = {  
    .year = 1995,  
    .title = "Євгеній Онегін"  
};
```

або

```
struct book lib = {  
    .author = "Пушкін О.С.",  
    .title = "Євгеній Онегін",  
    1995  
};
```

У першому і в другому варіантах при ініціалізації зазначаються найменування полів через крапку перед іменем поля. При цьому порядок і кількість полів не мають значення. У третьому прикладі перші два поля вказані через імена, а останній ініціалізується за порядковим номером - *три*, який відповідає полю *year*.

У деяких випадках має сенс створювати структури, які містять в собі інші (вкладені) структури. Наприклад, при створенні простого банку даних про співробітників підприємства доцільно ввести, принаймні, дві структури. Одна з них буде містити інформацію про прізвище, ім'я та по батькові співробітника, а друга буде включати в себе першу з додаванням полів про професії та вік:

```
struct tag_fio {  
    char last [100];  
    char first [100];  
    char otch [100];  
};  
  
struct tag_people {  
    struct tag_fio fio; // вкладена структура  
    char job [100];  
    int old;
```

```
};
```

Розглянемо спосіб ініціалізації та доступ до полів структури *people* на наступному прикладі:

Лістинг 7.2. Робота з вкладеними структурами.

```
int main() {
    struct people man = {
        {"Іванов", "Іван", "Іванович"},
        "Електрик",
        50};
    printf ("П.І.Б.:%s%s%s\n", man.fio.last, man.fio.first, man.fio.otch);
    printf ("Професія:%s\n", man.job); printf ("Вік:%d\n", man.old);
    return 0;
}
```

У даному прикладі показано, що для ініціалізації структури всередині іншої структури слід використовувати додаткові фігурні дужки, в яких міститься інформація для ініціалізації полів прізвища, імені та по батькові співробітника. Для того щоб отримати доступ до полів вкладеної структури виконується спочатку звернення до неї по імені *man.fio*, а потім до її полів: *man.fio.last*, *man.fio.first* і *man.fio.otch*. Використовуючи дане правило, можна створювати багаторівневі вкладення для ефективного зберігання та обробки даних.

Структури, як і звичайні типи даних, можна передавати функції в якості аргументу. Наступний приклад демонструє роботу функції відображення полів структури на екрані:

Лістинг 7.3. Передача структур через аргументи функції.

```
#include <stdio.h>

struct tag_people {
    char name [100];
    char job [100];
    int old;
};

void show_struct (struct tag_people man);

int main () {
    struct tag_people person = {"Іванов", "Електрик", 30};
    show_struct (person);
    return 0;
}

void show_struct (struct tag_people man) {
    printf ("Ім'я: %s\n", man.name);
    printf ("Професія: %s\n", man.job);
    printf ("Вік: %d\n", man.old);
}
```

}

У наведеному прикладі використовується функція з ім'ям *show_struct*, яка має тип аргументу *struct tag_people* і змінну-структурного типу *man*. При передачі структури функції створюється її копія, яка доступна в тілі функції *show_struct* під ім'ям *man*. Отже, будь-які зміни полів структури з ім'ям *man* ніяк не вплинуть на зміст змінної структурного типу з ім'ям *person* в головній функції. Разом з тим іноді необхідно виконувати зміну полів переданої структури у функції і повертати змінені дані у функцію, яка робила виклик. Для цього можна задати функцію, яка буде повертати структуру, як показано в лістингу 7.4:

Лістинг 10.4. Функції, які повертають нові значення полів у переданій на обробку структурній змінній.

```
#include <stdio.h>

struct tag_people {
    char name [100];
    char job [100];
    int old;
};

void show_struct (struct tag_people man);

struct tag_people get_struct();

int main() {
    struct tag_people person;
    person = get_struct();
    show_struct (person);
    return 0;
}

void show_struct (struct tag_people man) {
    printf ("Ім'я: %s\n", man.name);
    printf ("Професія: %s\n", man.job);
    printf ("Вік: %d\n", man.old);
}

struct tag_people get_struct() {
    struct tag_people man;
    scanf ("%s", man.name);
    scanf ("%s", man.job);
    scanf ("%d", man.old);
    return man;
}
```

У даному прикладі використовується функція *get_struct()*, яка працює з проміжною глобальною структурною змінною з іменем *man*, тип якої повністю збігається з типом локальної структурної змінної *person*. Функція запитує у користувача значення її полів і повертає введену інформацію головній

функції. В результаті виконання оператора присвоювання структури *man* структурі *person*, відбувається копіювання інформації відповідних полів проміжної змінної в локальну змінну.

Функціям в якості аргументу можна також передавати масиви структур. Для цього використовується наступне визначення заголовка функції:

```
void show_struct (struct people mans[], int size);
```

Тут *size* - число елементів масиву, яке необхідне для коректного зчитування інформації масиву *mans*. Наступний приклад показує принцип роботи з масивами структур:

Лістинг 7.5. Передача масиву структур до функції.

```
#include <stdio.h>
```

```
#define N 2
```

```
struct tag_people {  
    char name [100];  
    char job [100];  
    int old;  
};
```

```
void show_struct (struct people mans[], int size);
```

```
int main () {  
    struct people persons [N] = {  
        {"Іванов", "Електрик", 35},  
        {"Петров", «Викладач», 50},  
    };  
    show_struct (persons, N);  
}
```

```
void show_struct (struct people mans[], int size) {  
    for (int i = 0; i < size; i++) {  
        printf ("Ім'я: %s\n", mans[i].name);  
        printf ("Професія: %s\n", mans[i].job);  
        printf ("Вік: %d\n", mans[i].old);  
    }  
}
```

При передачі аргументу *persons* виконується копіювання інформації в масив *mans* і вказується додатковий параметр *size*, для визначення розміру масиву *mans*. Потім у функції *show_struct()* реалізується цикл, в якому виконується відображення інформації масиву структури на екрані монітора.

Робота з файлами

При роботі з файлами з використанням мови C можна виділити чотири основні дії:

1. Спочатку необхідно оголосити *файловий дескриптор*. Формат такого оголошення:

FILE ім'я_файлового_дескриптора;*

Наприклад, *FILE* fp;*

Слово **FILE** є стандартним ім'ям структурного типу файлового дескриптора, оголошеного в заголовному файлі *stdio.h*. У структурі **FILE** міститься інформація, за допомогою якої ведеться робота з потоком, зокрема: показчик на буфер, вказівник (індикатор) поточної позиції в потоці і т.ін.

2. Наступний крок - *відкриття потоку*, яке проводиться за допомогою стандартної функції *fopen()*. Ця функція повертає значення показчика на виділений системою потік роботи з файлом, який повинен присвоюватися оголошеному раніше файловому дескриптору. Відповідний оператор має вигляд:

ім'я_файлового_дескриптора = fopen(ім'я_файлу, режим_відкриття);

Обидва параметри функції *fopen()* являють собою рядки символів, які можуть бути як константами, так і показчиками на символні масиви. наприклад:

fp = fopen ("test.dat", "r");

Тут «*test.dat*» - це ім'я фізичного файлу в поточному каталозі диска, з яким тепер буде пов'язаний потік з дескриптором *fp*. Параметр режиму *r* означає, що файл відкритий виключно для читання. Що стосується термінології, то можна вживати як вираз «відкриття потоку», так і вираз «відкриття файлу».

При роботі з текстовими файлами існують наступні режими відкриття потоку і відповідні їм параметри (таблиця нижче):

Таблиця 7.1 – режими відкриття текстового потоку і відповідні їх параметри

Параметр	Режим
<i>r</i>	Відкрити для читання даних
<i>w</i>	Відкрити (створити) для запису
<i>a</i>	Відкрити для додавання даних у кінець файлу

Відкриття вже існуючого файлу для запису веде до втрати попередньої інформації в ньому. Якщо такий файл ще не існував, то він буде створений. Відкривати для читання можна тільки існуючий файл. Потік може бути відкритий або для текстового, або для двійкового (бінарного) режиму обміну. **Текстовий файл**: це послідовність символів, яка ділиться на рядки спеціальними кодами - повернення каретки (код 13) і переведення рядка (код 10). Якщо файл відкритий у текстовому режимі, то при читанні з такого файлу комбінація символів у файлі «повернення каретки - переведення рядку» при читанні буде перетворена в один символ '\n' - перехід до нового рядка.

При записі у файл здійснюється зворотне перетворення. При роботі з

двійковим файлом ніяких перетворень символів не відбувається, тобто інформація переноситься з/в пам'яті на диск без будь-яких змін.

Зазначені вище параметри режимів відкривають текстові файли. Якщо потрібно вказати на двійковий файл, то до другого параметру функції *fopen()* додається літера **b**. Наприклад: *rb*, або *wb*, або *r+b*.

Якщо при відкритті потоку з будь-якої причини виникла помилка, то функція *fopen()* повертає значення константи *NULL*. Ця константа також визначена у файлі *stdio.h*. Помилка може виникнути через відсутність файлу, що відкривається на диску, браку місця в пам'яті, на диску, тощо. Тому обов'язково після виклику функції *fopen()* слід проконтролювати результат, який вона повертає. Рекомендується наступний спосіб відкриття потоків:

```
FILE* fp;
if (fp = fopen("test.dat", "r")) == NULL) {
    puts ("Не можу відкрити файл test.dat \n");
    return;
}
```

3. Запис і читання символів. Запис символу в потік проводиться функцією *putc()* з прототипом:

```
int putc (int ch, FILE* fptr);
```

Якщо операція пройшла успішно, то повертається записаний символ. У випадку помилки повертається константа *EOF*.

Зчитування символу з потоку, відкритого для читання, здійснюється функцією *getc()* з прототипом:

```
int getc (FILE* fptr);
```

Функція повертає значення зчитуваного з файлу символу. Якщо досягнуто кінець файлу, то повертається значення *EOF*. Зауважимо, що це відбувається лише в результаті читання коду *EOF*.

Історично склалося так, що *getc()* повертає значення типу *int*. Те ж можна сказати і про аргумент *ch* в описі функції *puts()*. Використовується ж в обох випадках тільки молодший байт аргументу. Тому тип аргументу функції *ch* може бути і типу *char*.

4. Закриття потоку (файлу) здійснює функція *fclose()*, прототип якої має вигляд:

```
int fclose (FILE* fptr);
```

Тут *fptr* є іменем дескриптора файлу, що закривається. Функція повертає нуль, якщо операція закриття пройшла успішно. Інше значення буде означати помилку.

Неформатоване введення/виведення.

Лістинг 7.6. Програма запису в файл послідовності символів, що вводяться з клавіатури. Ознакою завершення введення буде символ '*'.

// Запис символів у файл

```
#include <stdio.h>
#include <conio.h>
void main () {
    FILE* fp;
    char c;
    if ((fp = fopen("test.dat", "w")) == NULL) {
        puts ("Не можу відкрити файл test.dat! \n");
        return;
    }
    puts ("Введіть символи. Ознака кінця - *");
    while ((c = getc()) != '*')
        putc (c, fp);

    if (!fclose (fp)) {
        puts ("Не можу закрити файл test.dat! \n");
        return;
    }
}
```

В результаті роботи коду лістингу 7.6 на диску (в поточному каталозі) буде створений файл з ім'ям *test.dat*, який заповниться введеними символами. Символ '*' в файл не запишеться.

Лістинг 7.7. Файл, створений в результаті роботи попередньої програми, потрібно послідовно (посимвольно) прочитати і вивести на екран.

// Читання символів з файлу

```
#include <stdio.h>
#include <conio.h>

void main() {

    FILE* fp;
    char c;

    if ((fp = fopen("test.dat", "r")) == NULL) {
        puts ("Не вдалося відкрити файл test.dat! \n");
        return;
    }
}
```

```

    }

    while ((c = getc (fp)) != EOF)
        putchar (c);

    if (!fclose (fp)) {
        puts ("Не можу закрити файл test.dat! \n");
        return;
    }
}

```

Зв'язок між результатом роботи попередньої програми і даною програмою здійснюється через ім'я фізичного файлу, яке в обох випадках має бути одним і тим самим.

Інший спосіб, з використанням функції *fgets()* дозволяє прочитати вміст файлу *по рядках*. Прототип функції *fgets()* має вигляд:

```
char* fgets(char *str, int num, FILE *stream)
```

Функція *fgets()* зчитує до *num-1* символів з файлу *stream* і розміщує їх в масиві символів, на який вказує *str*. Символи зчитуються до тих пір, поки не зустрінеться символ *новий рядок* '\n', *EOF* або до досягнення вказаної межі (*num-1*). Після закінчення зчитування в масив *str* відразу після останнього зчитаного символу поміщається нульовий символ завершення рядка '0'. Символ *новий рядок* при зчитуванні буде також збережений і стане частиною масиву *str*.

У разі успіху *fgets()* повертає *str*, при невдачі повертається *NULL*. Тому обов'язкова перевірка результату виконання до використання *str*! **У разі помилки зчитування вміст масиву, на який вказує *str*, не визначено.** Оскільки як в разі помилки, так і при досягненні кінця файлу повертається *NULL*, для визначення того, що саме сталося, необхідно використовувати функції *feof()* або *ferror()*.

Приклад використання *fgets()* наведено нижче:

Лістинг 7.8. Робота з функцією *fgets()*.

```

#include <stdio.h>
#include <conio.h>

void main () {

    FILE* fp;
    char S[30];
    int end_file = 0;

```

```

int ind_error;

if ((fp = fopen("test.dat", "r")) == NULL) {
    puts ("Не вдалося відкрити файл test.dat! \n");
    return;
}

while (!end_file) {
    ind_error = fgets (S, 30, fp);
    if (!ind_error)
        if (ferror()) {
            puts ("Помилка читання файла test.dat! \n");
            return;
        }
        else
            end_file = 1;
    fputs (S, stdout);
}

if (!fclose (fp)) {
    puts ("Не можу закрити файл test.dat! \n");
    return;
}
}

```

Функція *fgets()* зчитує з потоку *fp* в рядок *S* не більше 29 символів. Введення рядку закінчується при досягненні кінця рядка або максимальної кількості символів (у нашому випадку - 29). При досягненні кінця файлу функція *fgets()* повертає *NULL*. Функція *fputs()* виводить рядок у потік (у нашому прикладі на екран)

Форматоване введення / виведення.

За допомогою функцій форматного введення/виведення можна формувати на диску (або зчитувати з диска) текстовий файл з результатами обчислень, представленими в символьному вигляді аналогічно виведенню на екран. Надалі цей файл може бути переглянутий на екрані, роздрукований на принтері, відредагований за допомогою текстового редактора. Загальний вигляд функції **форматованого виведення**:

```
int fprintf (файловий_дескриптор, форматний_рядок, список змінних);
```

Використовувана нами раніше функція *printf()* для організації виведення на екран є варіантом функції *fprintf()*. Функція *printf()* працює лише зі стандартним потоком *stdin*, який за замовчанням пов'язується системою з дисплеєм. Не буде помилкою, якщо в програмі замість *printf()* написати

Правила використання специфікаторів форматів при записі в файли на диску точно такі ж, як і при виведенні на екран (дивитися конспект лекцій).

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

```
void main () {  
    FILE* fp;  
    int x;  
  
    if ((fp = fopen("test.dat", "w")) == NULL) {  
        puts ("Не вдалося відкрити файл test.dat! \n");  
        return;  
    }  
  
    //Вивід на екран і в файл "шапки" таблиці  
    printf ("\t Таблиця квадратів \n");  
    fprintf (fp, "\t Таблиця квадратів \n");  
    printf ("\tx\t\t\square(x)\n");  
    fprintf (fp, "\tx\t\t\square(x)\n");  
  
    //Обчислення і виведення таблиці квадратів на екран і в файл  
    for (x = 1; x <= 10; x++) {  
        printf ("\t%\ft%fn", float(x), pow(double(x),2));  
        fprintf (fp, "\t%\ft%fn", float(x), pow(double(x),2));  
    }  
  
    if (!fclose (fp)) {  
        puts ("Не можу закрити файл test.dat! \n");  
        return;  
    }  
}
```

Якщо тепер за допомогою текстового редактора (наприклад, *notepad*) відкрити файл *test.dat*, то на екрані побачимо:

x	square(x)
1.000000	1.000000
2.000000	4.000000
3.000000	9.000000
4.000000	16.000000

5.000000	25.000000
6.000000	36.000000
7.000000	49.000000
8.000000	64.000000
9.000000	81.000000
10.000000	100.000000

Форматне введення з текстового файлу здійснюється за допомогою функції *fscanf()*, яка має наступний вигляд:

```
int      fscanf      (файловий_дескриптор,      форматний_рядок,
список_адрес_змінних);
```

Даною функцією зручно користуватися в тих випадках, коли вихідні дані заздалегідь підготовлюються у текстовому файлі.

У наступному прикладі числові дані з файлу *test.dat*, отриманого в результаті виконання попередньої програми, вводяться в числові масиви *X* і *Y*. Для контролю значення елементів масивів виводяться на екран. Попередньо за допомогою текстового редактора у файлі *test.dat* видаляються два перші рядки з заголовками. В результаті у файлі залишаються тільки числа.

Лістинг 7.10: Програма, що вводить таблицю квадратів цілих чисел у масиви *X* та *Y* (виконує протилежну дію до програми у Лістингу 7.9). Для контролю ця таблиця виводиться ще і на екран.

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

void main () {
    FILE *fp;
    int i;
    double x[10], y[10];

    if ((fp = fopen("test.dat", "w")) == NULL) {
        puts ("Не вдалося відкрити файл test.dat! \n");
        return;
    }

    i = 0;
    while (feof (fp) == 0) {
        fscanf (fp, "%f%f", & x[i], & y[i]);
        printf ("%f%f \n", x[i], y[i]);
        i++;
    }

    if (!fclose (fp)) {
        puts ("Не можу закрити файл test.dat! \n");
        return;
    }
}
```

```

    }
}

```

Функція *fscanf()* зручна також для читання текстових файлів по словам (якщо роздільником слів є пробіл чи табуляція).

Функцію *fscanf()* зручно використовувати для читання з текстових файлів окремих записів в масив структур по окремим полям (якщо роздільником полів буде слугувати встановлений програмістом символ). Це досягається за допомогою наступного специфікатора у функції *fscanf()*:

fscanf (<дескриптор_файла>, "%[^<символ_обмежувач>]s", str);

При необхідності зчитування поточного запису із файлу з даними, який складається з послідовних полів деякої структури рекомендується зчитувати не увесь запис повністю, а окремими полями за один виклик функції *fscanf()*. Це дасть змогу фіксувати помилки читання (тим самим, і правильність введення інформації до файлу) на рівні окремих полів.

7.2. Завдання на комп'ютерний практикум

1. Перед виконанням завдання погодити з викладачем зміст полів структури, яку збирається використовувати для своєї задачі.
2. Написати програму роботи з масивом структур у відповідності з номером свого варіанту. Введення даних виконувати з текстового файлу. Повинна бути можливість введення імені файлу як в діалозі з користувачем, так і параметром в командному рядку при виклику програми. При виведенні даних у файл виконувати паралельне виведення для контролю і на екран. В якості імені текстового файлу для виведення взяти те саме ім'я, що і для введення, але з іншим розширенням.
3. Зробити висновки за отриманими результатами роботи програми.

7.3. Варіанти завдань

Варіант	Завдання
1	Написати програму пошуку книги по року видання у файлі
2	Написати програму видалення книги з файлу по прізвищу автора
3	Написати програму пошуку книг із заданим роком видання у файлі
4	Написати програму сортування книг у файлі за спаданням по року видання

Варіант	Завдання
5	Написати програму додавання нової книги у файл в лексикографічному порядку за прізвищем автора.
6	Написати програму пошуку співробітників з вказаним ім'ям (прізвищем) у файлі співробітників
7	Написати програму сортування студентів за зростанням шифрів їх груп у файлі
8	Написати програму видалення інформації про співробітника(-ів) з вказаним віком із файлу
9	Написати програму сортування книг у файлі по зростанню їх цін
10	Написати програму видалення установ з вказаною назвою із файлу установ
11	Написати програму видалення автомобілів із зазначеним роком виготовлення із файлу
12	Написати програму сортування готелів у файлі по зростанню кількості їхніх зірок
13	Написати програму сортування університетів у файлі по спадаючій за кількістю студентів
14	Написати програму пошуку списку квартир із заданою площею (від 1 до m^2) у файлі квартир
15	Написати програму сортування літаків за зростанням їх пасажиромісткості у файлі літаків

Варіант	Завдання
16	Написати програму видалення підприємств з негативним прибутком із файлу підприємств
17	Написати програму пошуку країн з кількістю населення більшою за задану у файлі країн
18	Написати програму пошуку населеного пункту з вказаною назвою у відповідному файлі
19	Написати програму пошуку шкіл у зазначеному населеному пункті у відповідному файлі
20	Написати програму сортування стадіонів у файлі стадіонів по країні і місткості стадіону
21	Написати програму пошуку товару по назві у файлі товарів
22	Написати програму видалення інформації із файлу про співробітника з вказаним віком та з членством у визначеній групі
23	Написати програму сортування у файлі товарів по спаданню їх цін
24	Написати програму пошуку у файлі користувача соціальної мережі за вказаним нікнеймом
25	Написати програму пошуку у файлі пацієнтів з інвалідністю, з відсортованим за алфавітом прізвищем
26	Написати програму видалення з файлу товарів, термін придатності у яких минув,

Варіант	Завдання
27	Написати програму пошуку у файлі студентів з успішністю, не нижче від заданої.
28	Написати програму сортування студентів за номером курсу у файлі за спаданням курсу і прізвищем
29	Написати програму пошуку співробітників з вказаною кваліфікацією у файлі співробітників
30	Написати програму пошуку у файлі студентів, що проживають в гуртожитку та виведення інформації про них в алфавітному порядку за зростанням

7.4. Зміст звіту

1. Титульний лист з назвою комп'ютерного практикуму, номером варіанту, прізвищем студента та групи.
2. Мета комп'ютерного практикуму.
3. Короткі теоретичні відомості.
4. Завдання.
5. Схеми алгоритмів.
6. Текст програм.
7. Вміст вхідних файлів для контрольних прикладів.
8. Результати роботи програм.
9. Висновки про отримані результати роботи програм.

7.5. Контрольні запитання

1. Запишіть структуру для зберігання імені, віку та місця роботи працівника.
2. Як задаються змінні на структурі?
3. Як задаються масиви структур?
4. Запишіть ініціалізацію структури зберігання книг.
5. Яким чином передаються структури функціям?
6. Чи можна передавати функції масив структур?

ПЕРЕЛІК ПОСИЛАНЬ

1. Шилдт Г. Полный справочник по Си (2004)
2. Прата С. Язык программирования С. Лекции и упражнения, 6-е изд. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2015. — 928 с.
3. Шпак З.Я. Програмування мовою С. – Львів: Оріяна-Нова, 2006. – 432 с.
4. ДСТУ 3008-2015. ЗВІТИ У СФЕРІ НАУКИ І ТЕХНІКИ. Структура і правила оформлювання (2015).
5. ГОСТ 19.701-90 "ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения".
6. Перри Г., Миллер Д. Программирование на С для начинающих, 3-е издание: Пер. с англ. – М.: ЭКСМО, 2015. - 369
7. Рендольф Ник и др. Visual Studio 2010 для профессионалов.: Пер. с англ. — М.: ООО "И.Д. Вильямс", 2011. — 1184 с.
8. Голуб А. И. Веревка достаточной длины, чтобы... выстрелить себе в ногу. Правила программирования на С и С++, 2-е изд. – Москва: McGraw–Hill, 2001. – 241 с.
9. Столлман Р. Стандарт кодирования GNU, 1994. Джерело http://www.opennet.ru/docs/RUS/coding_standard/
10. МакГрат М. Программирование на С для начинающих, 4-е издание.: Пер. с англ. – Москва: Эксмо, 2016. – 192 с.
11. Дейтел П.Дж., Дейтел Х.М. С для программистов с введением в С11. – Москва: ДМК Пресс, 2014. – 546 с.
12. Войтенко В.В., Морозов А.В. - С, С++. Теорія та практика, видання 2. – Житомир: ЖДТУ, 2004. – 324 с.
13. Вінник В.Ю. Алгоритмічні мови та основи програмування. Мова С. – Житомир: ЖДТК, 2007. – 328 с.
14. Керніган Б., Річі Д. Мова програмування С, друге видання. – 232 с.
15. Щедріна О. І. Алгоритмізація та програмування процедур обробки інформації С++. – Київ: КНЕУ, 2001. – 240 с.

ДОДАТОК 1. ВИМОГИ ДО ОФОРМЛЕННЯ КОДУ ПРОГРАМ НА МОВІ СІ

Угоди по ідентифікаторам

Підбір ідентифікаторів

А. Всі ідентифікатори повинні вибиратися з міркувань читабельності і максимального семантичного навантаження.

Наприклад:

```
const float Eps = 0.0001; // Точність
unsigned short Sum; // Сума
unsigned char Message [20]; // повідомлення
```

Невдалими можна вважати ідентифікатори:

```
const float UU = 0.0001; // Точність
unsigned short Kk; // Сума
unsigned char Zz [20]; // повідомлення
```

Б. Ідентифікатори рекомендується підбирати з слів англійської мови.
наприклад:

```
// Вивдає звуковий сигнал заданої частоти і тривалості
void Beep (unsigned short Hertz, unsigned short MSec);
```

```
// Вивдає True (1), якщо файл з ім'ям FName існує
unsigned char ExistFile (unsigned char * FName);
```

```
// Ознака закінчення роботи з програмою
unsigned char Done;
```

```
// Розміри виробу (ширина, висота)
unsigned short Width, Height;
```

Не дуже вдалим можна вважати ідентифікатори:

```
// Вивдає звуковий сигнал заданої частоти і тривалості
void Zvuk (unsigned short Chast, unsigned short Dlit);
```

```
// Вивдає True (1), якщо файл з ім'ям Im існує
unsigned char EstFile (unsigned char * Im);
```

```
// Ознака закінчення роботи з програмою
unsigned char Konec;
```

```
// Розміри виробу (ширина, висота)
unsigned short Shirina, Vysota;
```

Написання ідентифікаторів

Існує два основних способи написання ідентифікаторів.

А. У будь-якому ідентифікаторі кожне слово, що входить в ідентифікатор, писати, починаючи з великої літери, інші літери - маленькі.

Наприклад:

```
float NextX, LastX; // Наступна і попередня ітерація
char BeepOnError; // Звуковий сигнал при
                  // неправильних діях користувача
unsigned char FileName [20];
```

Б. У будь-яких ідентифікаторах кожне слово, що входить в ідентифікатор, розділяти символом `_`, при цьому всі букви - маленькі.

Наприклад:

```
float next_x, last_x; // Наступна і попередня ітерація
char beep_on_error; // Звуковий сигнал при
                  // неправильних діях користувача
unsigned char file_name [20];
```

В. Для ідентифікаторів імен функцій прийнято використовувати перше слово ідентифікатора з маленької літери, а усі наступні – з великої. Символ `_` при цьому не застосовується.

Наприклад:

```
// Стандартна функція модуля Graph; видає опис
// помилки використання графіки по її коду
unsigned char * graphErrorMsg (short err_code);
```

Г. Усі літери ідентифікаторів макросів, макровизначень у програмі прийнято писати використовуючи виключно великі літери.

Наприклад:

```
#define SIZE 100
#define ABS(x) ((x)<0?-(x):(x))
...
...
double Arr(SIZE);
...
...
c = ABS(a-b);
```

Угоди по самодокументованих програмах

Коментарі

А. Коментарі в тілі програми слід писати виключно українською чи російською мовою і по суті так, щоб програміст, який не брав участі в розробці програми (але має досвід роботи на мовах Сі/Асемблер), міг без особливих зусиль розібратися в логіці програми, і, при необхідності, супроводжувати даний програмний продукт.

Б. Рекомендується коментарі в програмі писати після символів `//`, а `/*` і `*/`

використовувати при налагодженні програми як "заглушки" ділянок програмного коду.

Специфікація функцій

Для кожної користувальницької функції повинна бути описана у вигляді коментаря специфікація, що містить наступну інформацію:

- призначення функції;
- опис семантики параметрів-значень (параметрів, що передаються за значенням);
- опис семантики параметрів-змінних (параметрів, що передаються по посиланню);
- опис семантики, що повертається.

Наприклад:

```
//////////////////////////////// Gauss //////////////////////////////////
// Рішення системи лінійних алгебраїчних рівнянь
// Методом Гаусса.
//
// Вхід:
// A - матриця коефіцієнтів системи;
// B - стовпець вільних членів системи;
// Eps - точність обчислень.
//
// Вихід:
// X      - вектор рішення
// HasSolution - прапорець, що встановлюється в True, якщо рішення
//              системи існує, і в False у всіх інших випадках
// NumOfRoots - число коренів в рішенні системи, може приймати
//              значення:
//              0 - якщо рішення системи не існує,
//              MaxN - якщо рішення системи існує і єдино,
//              MaxInt - якщо існує нескінченна безліч рішень;
// Det - значення визначника матриці A
// AForReverse - нижня трикутна матриця, отримана з A
//              в результаті виконання прямого ходу алгоритму Гауса
// BForReverse - стовпець вільних членів, отриманий
//              з B в результаті виконання прямого ходу
//              алгоритму Гаусса
//
// Результат Gauss: 0 - успішно, 1 – помилка.
////////////////////////////////
```

```
unsigned char Gauss( Matrix A, Vector B, float Eps,
                    Vector* X, char* HasSolution, short* NumOfRoots,
                    float* Det, Matrix* AForReverse, Vector* BForReverse
                    ) {
    ...
    return 0;
    ...
    return 1;
}
```

}

Примітка:

Якщо функція реалізує будь-який обчислювальний метод (наприклад: знаходження площі фігури методом трапецій, пошук мінімуму функції методом Ньютона і т.і.), рекомендується в тілі функції розмістити коментар з коротким описом методу, або посилання на джерело, де описаний метод.

Специфікація програмного файлу або модуля

Програмний файл або модуль повинен починатися зі специфікації у вигляді коментаря, що містить наступну інформацію:

- ідентифікація проекту, до якого належить файл;
- призначення (назва) та ім'я файлу;
- версія файлу;
- прізвище автора;
- опис модуля;
- історія змін модуля.

Наприклад:

```
/*-----
Проект:      ABC-1.0
Назва:       Математичні розрахунки
Файл:        primes.c
Версія:      1.0.2
Автор:       Наливайко І.М.
Опис:        Підрахунок кількості простих чисел.
Зміни:
-----
№  Дата      Версія  Автор          Опис
-----
1   01.03.16   1.0.1   Наливайко І.М.. Розрахунок на проміжку [1..200].
2   05.07.16   1.0.2   Підрахуйко І.М.  Розрахунок на проміжку
користувача.
-----*/
```

Примітка:

Після специфікації програмного файлу рекомендується помістити коментар до вказівок щодо запуску програми і роботі з нею (вказівками щодо використання модуля іншими програмістами) або посилання на джерело, яке використано при написанні програми (модуля).

Угоди по читабельності програм

"Драбинка"

"Драбинка" повинна відображати структурну вкладеність мовних конструкцій. Рекомендується відступ не менше 2-х і не більше 4-х пропусків (пробілів). Прийнятого відступу потрібно дотримуватися у всьому тексті програми. Правила написання деяких конструкцій:

```
if ( <умова> ) {
    <оператори>
}
```

```

if ( <умова> )
    <оператор>;

if ( <умова> ) {
    <оператори>
}
else {
    <оператори>
}

while ( <умова> ) {
    <операторы>
}

while ( <умова> )
    <оператор>;

for ( <ініц.. лічильника>; <умова>; <зміна лічильника> ) {
    <оператори>
}

for ( <ініц.. лічильника>; <умова>; <зміна лічильника> ) {
    <оператор>;
}

switch ( <вираз> ) {
    case <вираз>:
        <оператори>;
        break;
    .....
    default:
        <оператори>;
}

short Sign( float X ) {
    // повертає знак числа X
    if ( X > 0 )
        return 1;
    else if ( X < 0 )
        return -1;
    else
        return 0;
}

void Equation( float A, float B, float C,
               float* X1, float* X2, char* Num ) {
    // знаходження дійсних коренів квадратного рівняння;
    // A, B, C -- коефіцієнти
    // X1, X2 -- корені (якщо дійсного розв'язання не має, то
    // прирівнюються до 0);
    // Num    -- кількість коренів (0, 1, чи 2)

```

```

float D;
D = sqrt( B ) -4 * A * C;
if ( D < 0 ) {
    *Num = 0;
    *X1 = 0;
    *X2 = 0;
}
else {
    *X1 = ( -B + sqrt( D ) ) / ( 2 * A );
    *X2 = ( -B - sqrt( D ) ) / ( 2 * A );
    if ( *X1 == *X2 )
        *Num = 1;
    else
        *Num = 2;
}
}

```

Довжина рядків програмного тексту

Довжина рядків програми не повинна перевищувати ширини екрану (80 символів).

Інші рекомендації

А. Рекомендується операнди бінарних операцій (+, = та інші) відокремлювати від знака операції одною прогалиною.

Наприклад:

```
Sum = A + B;
```

Б. Рекомендується при перерахуванні ідентифікаторів після коми ",", ставити одну прогалину.

Наприклад:

```
printf ( "Сума:% d; Різниця:% d.", A + B, A - B );
unsigned short Day, Month, Year;
unsigned char i, j, k, l, m, n;
```

В. Рекомендується завжди писати символ-роздільник операторів ";" безпосередньо після оператора.

Наприклад:

```
switch (Num) {
    case 1:
        printf ( "один ...");
        break;
    case 2:
        printf ( "два ...");
        break;
```

```
case 3:
    printf ( "mpu ...");
    break;
default:
    printf ( "багато!");
}
```

Г. Рекомендується шістнадцяткові числа писати великими літерами.
наприклад:

```
#define BadDate 0xFFFF
#define kbEnter 0x0D // код клавіші <Enter>
```