МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи
«Функції форматного введення і виведення мови С»

з курсу «Програмування»
для студентів напряму 6.040302 – Інформатика
і курсу «Програмування та алгоритмічні мови»
для студентів напряму 6.040303 – Системний аналіз

Затверджено редакційно-видавничою
радою університету,
протокол № 1 від 04.06.14.

Харків
НТУ «ХПІ»
2014

Методичні вказівки до лабораторної роботи «Функції форматного введення і виведення мови С» з курсу «Програмування» для студентів напряму 6.040302 – Інформатика і курсу «Програмування та алгоритмічні мови» для студентів напряму 6.040303 – Системний аналіз / Уклад. М. І. Безменов, О. М. Безменова. – Х. : НТУ «ХПІ», 2014. – 17 с.

Укладачі: М. І. Безменов,
О. М. Безменова

Рецензент Л. М. Любчик

Кафедра системного аналізу і управління

ВСТУП

До складу стандартної бібліотеки мови С (і, як наслідок, С++) включені функції форматного введення/виведення. Використання таких функцій дозволяє вводити і виводити дані в заданому форматі, а також здійснювати елементарний аналіз уведеної інформації уже на етапі її читання. Функції форматного введення/виведення призначені для введення і виведення окремих символів, рядків, цілих вісімкових, шістнадцяткових, десяткових чисел і дійсних чисел всіх типів.

Метою даної лабораторної роботи є освоєння можливостей функцій форматного введення і виведення мови С.

1. ТЕОРЕТИЧНІ ОСНОВИ

1.1. Форматне виведення засобами мови С

Для роботи із стандартними потоком у режимі форматного введення використовуються описані в заголовному файлі stdio.h функції `printf()` і `printf_s()`, які мають такий формат:

```
int printf(форматний_рядок, список_аргументів);
int printf_s(форматний_рядок, список_аргументів);
```

Ці функції перетворюють дані із внутрішнього подання в символьний вигляд відповідно до форматного_рядка і виводить їх у вихідний потік. Дані, які перетворюються і виводяться є значеннями виразів, особливими випадками яких є константи і змінні, задаються в списку аргументів з використанням як роздільника символу «кома».

Функції повертають кількість виведених символів, а в разі помилки – від’ємне число.

Форматний_рядок може бути заданий або рядковою константою в літеральному вигляді (послідовністю символів, укладеною в подвійні лапки), або покажчиком на початок області пам’яті, у якій записано текстовий рядок (статичним або динамічним символьним масивом, останнім символом якого є символ `'\0'`). Віна може включати довільний текст, керуючі символи і специфікації перетворення даних. Виконання функцій `printf()` і `printf_s()` зводиться до сканування форматного рядка і

виведення даних відповідно до його вмісту. Текст і керуючі символи з форматного рядка просто копіюються у вихідний потік. Якщо ж при скануванні форматного рядка зустрічається специфікація перетворення даних, то здійснюється виведення відповідного аргументу зі списку аргументів із попереднім перетворенням значення, що виводиться, у відповідності зі специфікацією перетворення.

У списку аргументів їх кількість і типи повинні відповідати послідовності специфікацій перетворення у форматному рядку. Для кожного аргументу повинна бути вказана тільки одна специфікація перетворення. Якщо аргументів більше, ніж вказано у форматному рядку, «зайві» аргументи ігноруються. Список аргументів (з попередньою комою) може бути відсутнім. У цьому випадку здійснюється обробка (виведення) форматного рядка.

Специфікація перетворення має наступну форму:

% прапори ширина_поля.точність модифікатор специфікатор

Символ % є ознакою специфікації перетворення. У специфікації перетворення обов'язковими є тільки два елементи – ознака % і специфікатор.

Специфікація перетворення **не повинна містити в собі пробіл**и (у наведеній вище формі специфікації перетворення пробіли вставлені тільки для наочності). **Пробіл може зустрітися тільки** як прапор. Кожен елемент специфікації є поодиноким символом або числом.

Специфікації перетворень повинні відповідати типу вказаних аргументів. При розбіжності в типах повідомлення про помилки не видаються ні під час компіляції, ні під час виконання програми, але результат виведення може містити «сміття».

Елемент специфікації перетворення, названий специфікатором, визначає, як буде інтерпретуватися відповідний аргумент – як символ, як рядок або як число (див. табл.1.1).

Таблиця 1.1 – Деякі специфікатори форматного рядка функцій виведення

Специфікатор	Тип аргументу	Формат виведення
1	2	3
d, i	**int**, **char**, **unsigned**	Десяткове ціле зі знаком

Продовження табл. 1.1

1	2	3
u	**int**, **char**, **unsigned**	Десяткове ціле без знака
o	**int**, **char**, **unsigned**	Вісімкове ціле без знака
x, X	**int**, **char**, **unsigned**	Шістнадцяткове ціле без знака; при виведенні використовуються символи 0...9a...f (для специфікатора x) або символи 0...9A...F (для специфікатора X)
f	**double**, **float**	Дійсне значення зі знаком у вигляді: *знак_числа*dddd.dddd, де dddd – одна або більше десяткових цифр. Кількість цифр перед десятковою крапкою залежить від величини виведеного числа, а кількість цифр після десяткової крапки – від необхідної точності. *Знак_числа* за відсутності модифікатора + зображується тільки для від'ємного числа
e, E	**double**, **float**	Дійсне значення зі знаком у вигляді: *знак_числа*m.dddde*знак_порядка*xxx, де m.dddd – зображення мантиси числа (m – одна десяткова цифра; dddd – послідовність десяткових цифр); e – ознака порядку; xxx – десяткові цифри для подання порядку числа. *Знак_числа* за відсутності модифікатора + зображується тільки для від'ємного числа. Регістр у записі ознаки порядку залежить від регістру, у якому записаний специфікатор (e або E)
g, G	**double**, **float**	Дійсне значення зі знаком виводиться у форматі специфікаторів f або e (E) залежно від того, який з них більш компактний для даного значення і точності. Формат специфікатора e (E) використовується тоді, коли значення порядки менше −4 або більше заданої точності. Кінцеві нулі відкидаються, а десяткова точка з'являється, якщо за нею йде хоча б одна цифра. Регістр у записі ознаки порядку залежить від регістру, у якому записаний специфікатор (g або G)
c	**int**, **char**, **unsigned**	Поодинокий символ
s	**char** *	Символьний рядок. Символи виводяться або до першого нульового символу ('\0'), або ж виводиться та їх кількість, яку задано у полі точність специфікації перетворення
p	**void** *	Значення адреси. Друкує адресу, що задана аргументом (подання адреси залежить від реалізації)
%%		Виводить знак відсотка

Необов'язкові елементи специфікації перетворення (прапори, ширина поля та ін.) управляють іншими параметрами форматування.

Прапори керують вирівнюванням виводу і друком знака числа, пробілів, десяткової крапки, префіксів вісімкової і шістнадцяткової систем числення. Прапори можуть бути відсутні, а якщо вони є, то можуть стояти в будь-якому порядку. Значення прапорів наведено в табл. 1.2.

Таблиця 1.2 – Прапори форматування для функцій виведення

Прапор	Значення прапора
−	Виведене зображення значення притискається до лівого краю поля. За умовчанням (за відсутності цього прапора), відбувається вирівнювання по правій межі поля
+	Якщо виведене значення має знак + або −, то він виводиться. Без цього прапора знак виводиться тільки при від'ємному значенні
Пробіл	При виведенні додатного числа замість знака + виводиться пробіл
#	Якщо цей прапор використовується з форматами `o`, `x` або `X`, то будь-яке ненульове значення виводиться з попередніми `0`, `0x` або `0X` відповідно. При використанні прапора # з форматами `f`, `g`, `G` десяткова точка буде виводитися, навіть якщо в числі немає дробової частини

Ширина_поля задається цілим додатним числом і визначає мінімальну кількість позицій, яка відводиться для подання значення, що виводиться. Якщо число символів у значення, що виводиться, менше, ніж ширина_поля, здійснюється доповнення пробілами до заданої мінімальної довжини. Якщо ширина_поля задана з початковим нулем, не зайняті значущими цифрами позиції поля виведення зліва заповнюються нулями.

Якщо число символів у зображенні значення, що виводиться більше, ніж це визначено в ширині поля, або ширина поля не вказана, друкуються всі символи значення, що виводиться.

Точність задає

- мінімальну кількість цифр, які можуть бути виведені при використанні специфікаторів `d`, `i`, `o`, `u`, `x` або `X`;
- число цифр, які будуть виведені після десяткової точки при специфікаторах `e`, `E` і `f`;
- максимальне число значущих цифр при специфікаторах `g` і `G`;
- максимальну кількість символів, які будуть виведені при використанні специфікатора `s`.

Точність вказується за допомогою точки, за якою може йти невід'ємне ціле число в десятковому запису. Число після крапки є необов'язковим. Його відсутність є еквівалентною задаванню 0).

Слідом за точністю може бути вказаний модифікатор, який уточнює тип очікуваного аргументу. Він задається наступними символами:

- h – указує, що наступний після h специфікатор d, i, o, x або X застосовується до аргументу типу **short** або **unsigned short**;
- l – указує, що наступний після l специфікатор d, i, o, x або X застосовується до аргументу типу **long** або **unsigned long**;
- L – указує, що наступний після L специфікатор e, E, f, g або G застосовується до аргументу типу **long double**.

Особливістю функції printf_s() є те, що вона перевіряє форматний рядок на наявність тільки допустимих символів форматування, на відміну від функції printf(), яка таку перевірку не виконує.

1.2. Форматне введення засобами мови С

Форматне введення із вхідного потоку здійснюється функціями scanf() і scanf_s(), які описані в заголовному файлі stdio.h. Формат цих функцій аналогічний формату функцій printf() і printf_s():

```
int scanf (форматний_рядок, список_аргументів);
int scanf_s (форматний_рядок, список_аргументів);
```

Йі функції читають послідовності символів (байт) із вхідного потоку і інтерпретують їх відповідно до форматного_рядка як цілі числа, дійсні числа, одиночні символи або рядки.

Після перетворення у внутрішнє подання дані записуються в області пам'яті, визначені аргументами, які слідують за форматним_рядком. Кожен аргумент **повинен** визначати адресу змінної, у яку буде записано чергове значення, що вводиться, і тип якої відповідає типу, зазначеному у специфікації перетворення з форматного_рядка. Адреса змінної отримується застосуванням операції & до імені змінної (наприклад, адреса змінної w записується як &w).

Якщо аргументів більше, ніж потрібно у форматному рядку, «зайві» аргументи ігноруються. Якщо аргументів недостатньо для даного форматного рядка, то результат залежить від операційної системи і компілятора.

Послідовність кодів символів, яку функції форматного введення читає із вхідного потоку, як правило, складається з полів (рядків), розділених будь-якою кількістю пробілів або узагальнених пробільних символів. Поля проглядаються і вводяться цими функціями посимвольно. Якщо у специфікації перетворення вказано точну кількість символів, що вводяться, то введення поля припиняється при її досягненні.

Функції функціями `scanf()` і `scanf_s()` завершують роботу, якщо вичерпується форматний рядок. При успішному завершенні функція повертає кількість об'єктів, які отримали значення при введенні. У разі виникнення ситуації «кінець файлу» повертається значення `EOF`, а при виникненні помилки перетворення даних – значення `-1`.

Форматний рядок може включати:

- пробільні символи, які відстежують розділення вхідного потоку на поля. Пробільний символ указує на те, що із вхідного потоку треба зчитувати (але не зберігати) усі послідовні пробільні символи аж до появи непробільного символу. Один символ пробілу у форматному рядку відповідає будь-якій кількості послідовних пробільних символів у вхідному потоці;
- звичайні символи, відмінні від пробільних і символу `%`. Обробка звичайного символу з форматного рядка зводиться до читання чергового символу із вхідного потоку. Якщо прочитаний символ відрізняється від оброблюваного символу форматного рядка, наступні за ним вхідні символи залишаються непрочитаними;
- специфікації перетворення.

Звичайні символи у форматному рядку використовуються для контролю введення. Дані вводитимуться, якщо послідовність звичайних символів форматного рядка співпаде із послідовністю символів, починаючи з поточної позиції рядка введення. Таким чином, звичайні символи відіграють роль паролю, без уведення якого стає неможливим здійснення подальшого введення.

Специфікація перетворення має таку форму:

`% * ширина_поля модифікатор специфікатор`

Як і у випадку функцій форматного виведення специфікація перетворення **не повинна містити** всередині себе **пробілів** (у наведеній вище формі специфікації перетворення пробіли вставлені тільки для наочності).

Як і для `printf()` і `printf_s()`, у специфікації перетворення функцій `scanf()` і `scanf_s()` обов'язковими символами є символ `%`, який є ознакою специфікації перетворення, і специфікатор, який дозволяє вказати очікуваний тип елемента даних у вхідному потоці (табл. 1.3).

Символ `*`, указаний слідом за символом `%`, забороняє записування прочитаного значення у комірку пам'яті, що визначається поточним аргументом списку введення.

Ширина_поля – додатне десяткове ціле, що визначає максимальну кількість символів, які можуть бути прочитані із вхідного потоку. Фактично може бути прочитано менше символів, якщо зустрінеться символ пробілу або символ, який не може бути перетворений за заданою специфікацією.

Таблиця 1.3 – Основні специфікатори форматного рядка функцій введення

Специфікатор	Очікуваний тип даних, що вводяться	Тип змінної, що приймає значення
`d`	Десяткове ціле	**`int`**
`o, O`	Вісімкове ціле	**`int`**
`x, X`	Шістнадцяткове ціле	**`int`**
`i`	Десяткове, вісімкове або шістнадцяткове ціле	**`int`**
`u`	Десяткове ціле без знака	**`unsigned int`**
`e, E, f, g, G`	Дійсне значення виду: [+|-]dddd [E|e[+|-]dd], що складається з необов'язкового знаку + чи −, послідовності dddd із однієї або більшої кількості десяткових цифр з можливою десятковою точкою і необов'язкового порядку (ознака `e` або `E`, за якою йде ціле значення, можливо, зі знаком). Для введення значень змінних типу **`double`** використовуються специфікатори `%le`, `%lE`, `%lf`, `%lg`, `%lG`, а для введення значень змінних типу **`long double`** використовуються специфікатори `%Le`, `%lE`, `%Lf`, `%Lg`, `%lG`.	**`float`**
`c`	Символ. Черговий символ, що читається, завжди сприймається як значущий символ. Пропуск початкових пробільних символів у цьому випадку придушується. Для введення найближчого символу, відмінного від пробільного, необхідно використовувати специфікацію `%ls`.	**`char`**
`s`	Рядок символів, обмежений праворуч і ліворуч пробільними символами. Для читання рядків, не обмежених пробільними символами, замість специфікатора `s` слід використовувати набір символів у квадратних дужках. У цьому разі символи із вхідного потоку читаються до першого символу, відмінного від символів у квадратних дужках. Якщо ж першим символом у квадратних дужках заданий символ `^`, то символи із вхідного потоку читаються до першого символу з квадратних дужок.	Масив символів, достатній для розміщення вхідного рядка, плюс завершальний символ кінця рядка (`'\0'`), який додається автоматично

Модифікатор – дозволяє задати довжину змінної, у яку передбачається помістити значення, що буде вводитися. Можливі такі модифікатори:

L – означає, що аргумент, який відповідає специфікації перетворення, повинен указувати на об'єкт типу **long double**;

l – означає, що аргумент повинен указувати на змінну типу **long**, **unsigned long** або **double**;

h – означає, що аргумент повинен указувати на тип **short**.

Особливістю функції scanf_s() є наступне: якщо у форматному рядку зустрічається один з символів s, S, c, C (тобто при введенні рядків або символів), то у списку аргументів слідом за відповідним покажчиком на початок рядка або символьну змінну через кому необхідно вказати розмір буфера, який відводиться під елемент, що вводиться:

```
double v;
char c, s[80];
scanf_s("%Lf%c%s", &v, &c, 1, s, sizeof(s));
```

У разі використання функції scanf() компілятор виводить попередження про її небезпечність. Щоб заблокувати виведення цього попередження треба визначити ім'я _CRT_SECURE_NO_WARNINGS:

```
#define _CRT_SECURE_NO_WARNINGS
```

2. ПРИКЛАДИ ПРОГРАМ

Приклад 1. Приклади використання функції printf().

Розв'язок.

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

int main()
{
            // Використання printf() без списку аргументів:
    printf("Hello!\n");       // Після виведення здійсниться
                              // перехід до нового рядка - \n
    printf("My name is Olga.");      // Це - у новому рядку
/*
Результат виведення:
Hello!
My name is Olga.
*/
    int u = 5, v = 5, w = -9, r = -9;
     // Виведення символів форматного рядка і цілих змінних
            // (спочатку буде перехід до нового рядка - \n)
```

```
            // Зверніть увагу на наявність прапора "пробіл"!
    printf("\nu=%d, v=% d, w=%d, z=% d ", u, v, w, r);
/*
Результат виведення:
u=5, v= 5, w=-9, r=-9
*/
              // Виведення значення виразу і цілої константи
    printf("\n2u+v+w=%d, %d", 2 * u + v + w, -123);
/*
Результат виведення:
2u+v+w=25, -123
*/
    float x = 10.5, y = 130.67, z = -54;
        // Виведення дійсних значень у форматі з фіксованою
       // крапкою (дві цифри після десяткового роздільника)
    printf("\nCoordinates of the object: x:%.2f, "
           "y:%.2f, z:%.2f\n", x, y, z);
/*
Результат виведення:
Coordinates of the object: x:10.50, y:130.67, z:-54.00
*/
                    // Виведення тексту в лапках і символу %
    printf("\"Quoted text\"");
    printf("\nOxygen content: 100%%");
/*
Результат виведення:
"Quoted text"
Oxygen content: 100%
*/
    int k = 26;
       // Виведення цілого числа у шістнадцятковому поданні
    printf("\nk-dec=%d, k-hex=%X", k, k);
/*
Результат виведення:
k-dec=26, k-hex=1A
*/
                             // Виведення поодиноких символів
    char ch1 = 'A', ch2 = 'B', ch3 = 'C';
    printf("\n%c%c%c%c\n", ch1, ch2, ch3, 68);
/*
Результат виведення:
ABCD
```

```
*/
    char str[20] = "my string.";
    // Виведення вмісту форматного рядка і звичайного рядка
    printf("This is %s", str);
/*
Результат виведення:
This is my string.
*/
    printf("\nPress any key to exit.\n");
    _getch();
    return 0;
}
```

Приклад 2. Уведення цілого числа типу **int**.

Розв'язок.

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

int main()
{
    int age;
    printf("How old are you? ");
    scanf("%d", &age);
    printf("You %d years.", age);
    printf("\nPress any key to exit.\n");
    _getch();
   return 0;
}
```

Приклад 3. Програма-підсумовувач. Уводяться два цілі числа і обчислюється їхня сума. При введенні одразу за першим доданком потрібно набрати символ «+», не вводячи перед ним ніякого іншого роздільника. Далі набирається другий доданок і здійснюється введення. Наприклад, при введенні `100+34` буде виведений такий результат: `100+34=134`.

Розв'язок.

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

int main()
{
```

```
    int x, y;
    printf("Calculator: ");
    scanf("%d+%d", &x, &y);
    printf("%d+%d=%d", x, y, x + y);
    printf("\nPress any key to exit.\n");
    _getch();
    return 0;
}
```

Приклад 4. У програмі задається ширина поля зчитування (вона дорівнює 10 символам). Якщо ввести рядок з більшою кількістю символів, то всі символи після 10-го будуть відкинуті. Вони залишаться в буфері клавіатури і будуть читатися наступним оператором уведення (якщо він є в програмі). Відзначимо, що у виклику функції `scanf()` знак `&` не вказано перед іменем масиву `name`, оскільки ім'я масиву є адресою його першого елементу.

Розв'язок.

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

int main()
{
    char name[11];
    printf("Enter your login (no more than "
           "10 characters):");
    scanf("%10s", name);
    printf("\nYou entered %s", name);
    printf("\nPress any key to exit.\n");
    _getch();
    return 0;
}
```

Приклад 5. Використання множини пошуку. Читання здійснюється до першого з символів, відмінного від символів від 0 до 9. Потім зчитуються всі наступні символи рядка введення.

Розв'язок.

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

int main()
{
```

```
    char str1[100], str2[100];
    scanf_s("%[0123456789]%s", str1, sizeof(str1),
                               str2, sizeof(str2));
    /* Уведемо набір символів:
12345abcdefg456
    */
    printf("%s\n%s", str1, str2);
    /* На екран програма виведе:
12345
abcdefg456
    */
    printf("\nPress any key to exit.\n");
    _getch();
    return 0;
}
```

3. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

За час, відведений для виконання лабораторної роботи (2 академічні години), студент повинен:

1. Написати програми для розв'язання наведених нижче задач.
2. Здійснити налаштування програми, виправивши синтаксичні та логічні помилки.
3. Оформити звіт до лабораторної роботи.
4. Відповісти на контрольні запитання.

Примітка. Усі студенти розробляють програми для розв'язання однакових задач. Процес прийому лабораторної роботи передбачає можливість деякої модифікації умов для забезпечення деякої варіативності і перевірки самостійності виконання роботи.

4. ВАРІАНТИ ЗАДАЧ

1. Увести два цілих додатних числа *m* і *n*, записані у шістнадцятковому поданні у форматі 0хцифри. Вивести суму, різницю, добуток цих чисел, а також цілу частину і остачу від ділення *m* на *n*. При виведенні подати результати у десятковій, вісімковій і шістнадцятковій системах числення в супроводі відповідного повідомлення. Вісімкові й шістнадцяткові числа подавати у двох форматах – без ознаки основи системи числення (0 і 0x) та з її наявністю. Вивести також дійсне значення, що

дорівнює відсотковому відношенню числа *m* до числа *n*, подавши його у двох форматах – як число з фіксованою крапкою і як число із плаваючою крапкою. Продемонструвати використання ширини поля виведення, а також керування кількістю цифр після десяткової крапки.

2. Двічі ввести одне й те саме дійсне число із записуванням результату введення один раз у змінну типу **`double`**, а другий – у змінну типу **`float`**. У обох випадках обчислити синус уведеного значення (функція обчислення синуса оголошена у заголовному файлі math.h і має формат виклику `sin(x)`, де `x` – дійсний вираз, змінна або константа). Вивести результати обчислення у форматах із фіксованою і плаваючою крапкою без використання елементу специфікації «ширина поля». Порівняти результати виведення. Супроводжувати виведення пояснюючими повідомленнями.

5. КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як можна вивести текстовий рядок?
2. Яка функція здійснює форматне введення? Як здійснюється її виклик?
3. Опишіть сенс окремих елементів форматного рядка функції форматного введення.
4. Як задається специфікація перетворення при форматному введенню?
5. Що визначає ширина поля при форматному введенні?
6. Що визначає модифікатор у специфікації перетворення функції форматного введення? Які існують модифікатори і як вони задаються?
7. Як задаються окремі елементи списку введення?
8. Перелічіть основні специфікатори введення.
9. Що означають квадратні дужки як специфікатор уведення?
10. Чим відрізняється введення даних для змінних типу **`float`** від уведення даних у змінні типу **`double`**?
11. Нижче наведено текст програми:

```
#include <stdio.h>

int main ()
{
    int k;
    int ret;            // Код повернення функції scanf()
    char c, s[80];
    ret=scanf("code: %d %*s %c %s", &k, &c, s);
```

```
    printf("\n k=%d c=%c s=%s", k, c, s);
    printf("\n\t ret = %d\n", ret);
    return 0;
}
```

Наведіть результат її виконання для таких рядків уведення:

а) `code:  15  uvwXYZ  W  qwerty`

б) `code: -5 Z qwerty`

в) `cod:  5  uvwXYZ  Y  qwerty`

СПИСОК ЛІТЕРАТУРИ

1. Керниган, Б. Язык программирования Си / Б. Керниган, Д. Ритчи. – М. : Финансы и статистика, 1992. – 272 с.
2. Павловская, Т. А. С/С++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2003. – 461 с.
3. Подбельский, В. В. Программирование на языке Си / В. В Подбельский, С. С. Фомин. – М. : Финансы и статистика, 1999. – 600 с.
4. Страуструп, Б. Язык программирования Си++ : Второе издание / Б. Страуструп. – К. : ДиаСофт, 1993. – Ч. 1. – 264 с. ; Ч. 2. – 296 с.
5. Подбельский, В. В. Язык Си++ / В. В. Подбельский. – М. : Финансы и статистика, 1999. – 560 с.
6. Либерти, Дж. Освой самостоятельно С++ за 21 день : учеб. пособ. / Джесс Либерти. – М. : Вильямс, 2001. – 816 с.
7. Савитч, У. Язык С++. Курс объектно-ориентированного программирования / Уолтер Савитч. – М. : Вильямс, 2001. – 704 с.
8. Шилдт, Г. С++: руководство для начинающих / Герберт Шилдт. – М. : Вильямс, 2005. – 6г2 с.
9. Шилдт, Г. Самоучитель С++ / Г. Шилдт. – СПб. : BHV-Петербург, 2003. – 688 с.
10. Шилдт, Г. Полный справочник по С++ / Герберт Шилдт. – М. : Вильямс, 2006. – 800 с.

Навчальне видання

Методичні вказівки
до лабораторної роботи
«Функції форматного введення і виведення мови С»
з курсу «Програмування» для студентів напряму
6.040302 – Інформатика і курсу «Програмування
та алгоритмічні мови» для студентів напряму
6.040303 – Системний аналіз

Укладачі: БЕЗМЕНОВ Микола Іванович,
БЕЗМЕНОВА Ольга Миколаївна

Відповідальний за випуск О. С. Куценко
Роботу до видання рекомендував О. В. Горілий

За авторською редакцією

План 2014 р., поз. 111.

Підп. до друку 07.07.2014 р. Формат 60×84 1/16. Папір офсетний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 0,9. Наклад 50 пр.
Зам. № 243. Ціна договірна.

Видавець і виготовлювач
Видавничий центр НТУ «ХПІ»,
вул. Фрунзе, 21, Харків, 61002.

Свідоцтво суб'єкта видавничої справи ДК № 3657 від 27.12.2009 р.