МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ПОКАЖЧИКИ І МАСИВИ С \ С ++ У ПРИКЛАДАХ

Методичні вказівки для самостійної роботи з дисципліни
«Програмування і алгоритмічні мови»

для бакалаврів напряму підготовки 12 Інформаційні технології
та 17 Електроніка та телекомунікації

Дніпро
2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ ЕЛЕКТРОЕНЕРГЕТИКИ

Факультет інформаційних технологій

Кафедра безпеки інформації та телекомунікацій

ПОКАЖЧИКИ І МАСИВИ С \ С ++ У ПРИКЛАДАХ

Методичні вказівки для самостійної роботи з дисципліни
«Програмування і алгоритмічні мови»

для бакалаврів напряму підготовки 12 Інформаційні технології
та 17 Електроніка та телекомунікації

Дніпро
2020

Покажчики і масиви С \ С ++ у прикладах. Методичні вказівки до виконання лабораторних робіт з дисципліни «Програмування і алгоритмічні мови» для бакалаврів напряму підготовки 12 Інформаційні технології та 17 Електроніка та телекомунікації/ Г.М. Саксонов, О.А. Жукова, Сєчкін І.А. – Дніпро: НТУ «ДП», 2020. – 17 с.

Автори:
Г.М. Саксонов, О.А. Жукова, Сєчкін І.А.

Затверджено методичною комісією за напрямом Кібербезпека (протокол № 2 від.08.02.2020) за поданням кафедри безпеки інформації та телекомунікацій (протокол № 7 від 25.02.2020).

Відповідальний за випуск зав. кафедри БІТ В.І. Корнієнко, д-р техн. наук, проф.

ЗМІСТ

ПЕРЕДМОВА

Покажчики в мови С / С ++ в використовуються набагато інтенсивніше, аніж в інших мовах, тому що іноді деякі обчислення виразити можливо лише за їх допомогою, а частково й тому, що з ними утворюються більш компакті та ефективніші програми, аніж програми з використанням звичайних засобів. Існує твердження – аби стати знавцем С / С ++ , потрібно бути спеціалістом з використання покажчиків.

Отже ці методичні вказівки (автори мають таку надію) допоможуть з цим розібратися.

1. ЗАГАЛЬНІ ВИЗНАЧЕННЯ ТА ПОЗНАЧЕННЯ

Поняття змінної можна визначити як ім'я елементу пам'яті, в якому зберігається значення вказаного типу. Кожен елемент пам'яті має свою унікальну адресу.

На рис 1 показана умовна схема, на якому змінна ***X*,** що має значення **134**, а умовна адреса елементу пам'яті, де зберігається ця змінна, дорівнює **00AF.**

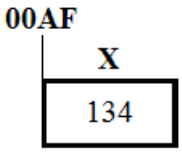

Рис 1. Умовне позначення змінної *X* в оперативній пам'яті

В мові С / С ++ розрізняють три види покажчиків - покажчики на об'єкт певного типу даних (в подальшому просто покажчиком), на функцію і безтіповий покажчик, що відрізняються властивостями і набором допустимих операцій. Покажчик не є самостійним типом даних, він завжди пов'язаний з будь-яким іншим конкретним типом даних.

З поняттям покажчик тісно пов'язана операція взяття адреси змінної.

Для визначення адреси змінної програма може використовувати операцію знаходження адреси (*&*).

Для виведення адреси змінної функцією *printf* використовується модифікатор *% p*.

Наприклад, в наступній програмі для знаходження адреси використовується операція *&*, а для виведення на екран адреси змінної функція *printf* с модифікатором *% d*:

```
#include <stdio.h>
#include <locale.h>
void main()
{           setlocale(LC_ALL,"Russian");
            int X=134;
      printf("Значення змінної X - %d\n", &count);
      printf("Адреса змінної X - %p\n", &salary);
}
```

Після компіляції і виконання цієї програми на екран буде виведено:

Значення змінної X - 134

Адреса змінної X - 00AF

Схема розташування змінної *X* в пам'яті показана на рис. 1.

2 .ОГОЛОШЕННЯ ПОКАЗЧИКІВ

Покажчик - це змінна, значеннями якої є адреси в оперативній пам'яті.

Для зберігання покажчиків в програмі необхідно оголосити змінну (змінна-вказівник). Для оголошення покажчика необхідно вказати тип значення, на яке вказує покажчик (наприклад, *int*, *float*, і т. п.) та поставити зірочку (*) перед ім'ям змінної.

Покажчик на об'єкт містить адресу області пам'яті, в якій зберігаються дані певного типу (основного або складеного). Найпростіше оголошення покажчика на об'єкт (в подальшому просто покажчик) має вигляд:

*тип * ім'я;*

Зірочка відноситься безпосередньо до імені. Для того, щоб оголосити кілька покажчиків, потрібно ставити її перед ім'ям кожного з них. Наприклад, в операторі *int * a, b, * c;* описуються два покажчика на ціле з іменами *a* та *c* а також ціла змінна *b*.

Покажчик може бути константою або змінною:

*int * pi*; // покажчик на цілу змінну.

*const int * pci;* // покажчик на цілу константу.

Після оголошення покажчика бажано форматувати його нулем *(NULL)*. Наприклад, в наступній програмі оголошується покажчик *int * ptr*.

```
#include <stdio.h>
#include <locale.h>
void main()
{
setlocale (LC_ALL,"Russian");
        int X = 134;
        int * ptr=0;
printf("Адреса змінної X - %p\n", &X);
printf("Значення змінної X - %d\n", X);
printf("Адреса змінної ptr - %p\n", &ptr);
printf("Значення змінної ptr - %p\n", &ptr);
}
```

Після компіляції і виконання програми на екран (див. рис. 2) буде виведено:

Адреса змінної X - 00AF

Значення змінної X - 134

Адреса змінної ptr - 00B4

Значення змінної ptr – 0000

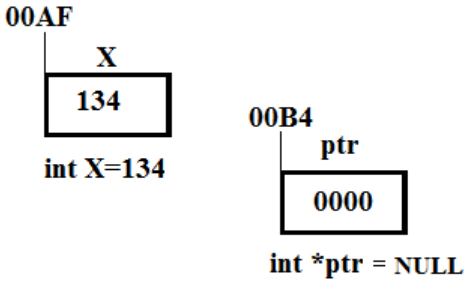

Рис 2. Умовне позначення значень і адрес змінної *X* і покажчика *ptr* в пам'яті

3. ІНІЦІАЛІЗАЦІЯ ПОКАЖЧИКІВ

При визначенні покажчика треба прагнути виконати його ініціалізацію, тобто присвоєння початкового значення. Ненавмисне використання неініціалізованих покажчиків – поширене джерело помилок в програмах. Існують наступні основні способи ініціалізації покажчика.

3.1 Присвоєння пустого значення

int* X = NULL;

int *Y = 0;

У першому рядку використовується константа *NULL*, яка визначається в деяких заголовочний файлах мови С як покажчик, значення якого дорівнює нулю. Оскільки гарантується, що об'єктів з нульовою адресою немає, порожній покажчик можна використовувати для перевірки того, чи посилається покажчик на конкретний об'єкт чи ні.

Приклад фрагменту програми з ініціалізацією нулем наведено вище.

3.2. Присвоєння покажчику адреси існуючого об'єкта за допомогою операції одержання адреси

Наприклад, в програмі оголошується покажчик *int * pt,* якому присвоєно адресу змінної *X:*

```
#include <stdio.h>
#include <locale.h>
void main()
{
setlocale(LC_ALL,"Russian");
    int X = 134;
    int* ptr=&X;
printf("Адреса змінної X - %p\n", &X);
printf("Значення змінної X - %d\n", X);
printf("Адреса змінної ptr - %p\n", &ptr);
printf("Значення змінної ptr - %p\n", &ptr);
}
```

У першому рядку використовується константа *NULL*, визначена в деяких заголовочний файлах С як покажчик, значення якого дорівнює нулю. Оскільки гарантується, що об'єктів з нульовим адресою немає, порожній покажчик можна використовувати для перевірки того, посилається покажчик на конкретний об'єкт чи ні. Приклад програми з ініціалізацією нулем наведено вище.

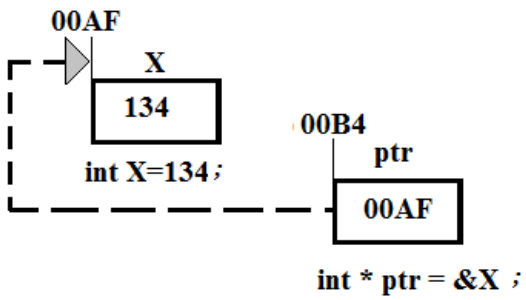

Рис 3. Умовне позначення адресів та значеньзмінної *X* і покажчика *ptr*

3.3. Ініціалізація іншим покажчиком

В наступній програмі оголошується покажчик *int * ptr,* якому було присвоєно адресу змінної *X* (див. рис. 4):

int X = 134;

int * ptr=&X;

int * pdd = ptr;

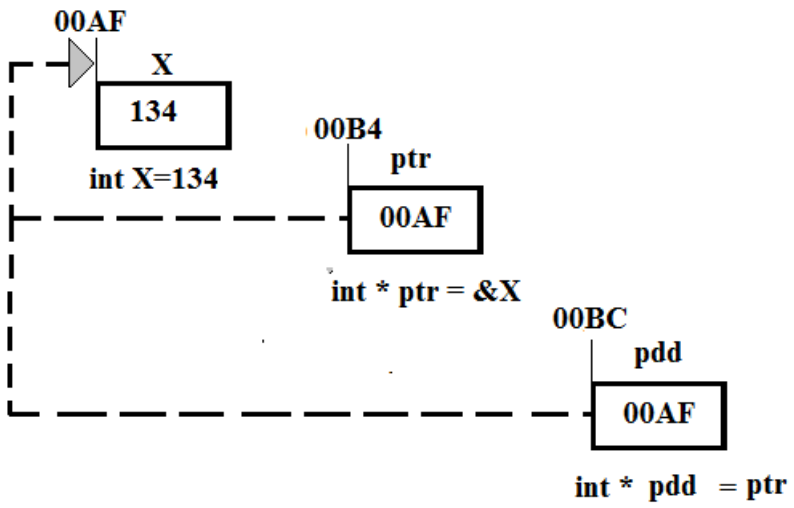

Рис 4. Умовна схема значень і адрес змінної *X* і покажчиків *ptr* і *pdd*

4. ОПЕРАЦІЇ З ПОКАЖЧИКАМИ

4.1 Операція розіменування покажчика

Операція розіменування (розадресації) покажчика призначена для доступу до комірки пам'яті, адреса якої зберігається в покажчику.

Операція розіменування покажчика *name* має вигляд * *name*. Цю дію можна висловити словами «значення за адресою, яка записана в *name*».

Цю операцію можна використовувати як для отримання, так і для зміни значення величини, на яку цей покажчик «вказує» (якщо вона не оголошена як константа).

4.1.1 Отримання значення величини, адреса якої зберігається в покажчику

Розглянемо фрагмент коду програми і схему її змінних (рис 5).

int X=134;

*int * ptr=&X;*

int Y= * ptr*; //** Набуття значення величини, адреса якої зберігається в покажчику ***ptr

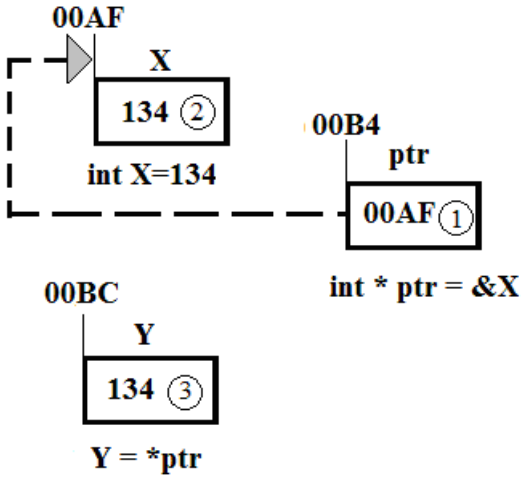

Рис 5. Умовна схема отримання значення величини *Y*

Оператор *int Y = * ptr* виконує дві операції:

- операцію розіменування покажчика;
- операцію присвоювання.

Операція розіменування має вищий пріоритет у порівнянні з операцією присвоєння, і тому виконується першою.

Дію оператора *int Y = * ptr* можна умовно уявити як послідовне виконання наступних дій:

- читати адресу, збережену в покажчику *ptr (00AF);*
- отримати значення, що знаходиться за цією адресою *(134);*
- присвоїти змінної *Y* результат попередньої дії.

4.1.2 Зміна значення величини, адреса якої зберігається в покажчику

Розглянемо фрагмент кода програми:

int X=134;

int Y=555;

*int * ptr=&X;*

**** ptr=Y;***

Умовна схема її змінних зображена на рис 5.

Оператор * *ptr* = *Y* послідовно виконує дві операції: операцію розіменування покажчика, а потім операцію присвоювання.

Операція розіменування покажчика має вищий пріоритет у порівнянні з операцією присвоєння, і тому виконується першої

Дія оператора * *ptr* = *Y* можна умовно уявити як послідовне виконання наступних дій:

- читати адресу, збережену в покажчику *ptr (00AF);*
- присвоїти змінної розташованої за цією адресою значення змінної Y;

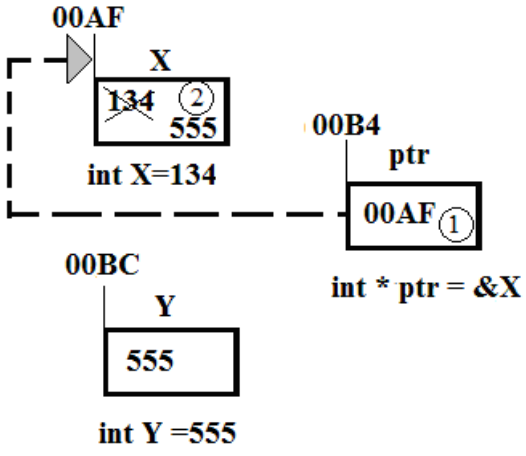

Рис 5. Умовна схема зміни величини, адреса якої зберігається в покажчику

4.2 Операції адресної арифметики

4.2.1 Операції підсумовування і віднімання цілих чисел

У мові C / C ++ допустимі арифметичні операції над покажчиками: підсумовування покажчика з цілим числом і віднімання з покажчика цілого числа. Результатом цих операцій буде нова адреса. Операція підсумовування покажчика з цілим числом «пересуває» покажчик в бік збільшення адреси, а операція віднімання – в бік зменшення адреси.

Для всіх покажчиків адреса збільшується або зменшується на величину, яка дорівнює розміру об'єкта того типу, на який вони вказують. Тому покажчик завжди посилається на об'єкт з типом, тотожним базового типу покажчика.

Розглянемо фрагмент коду програми:

float M= 0.567;

float N= -12.45;

float K= 109;

float L= 23.002;

float *ptr = &N;

ptr = ptr + 2;

ptr = ptr – 3;

Умовна схема зміни значень покажчика *ptr* при послідовному виконанні операцій додавання покажчика з цілим числом *3* і віднімання з нового значення покажчика числа *2*, показано на рис 6. При підсумовуванні значення покажчика збільшується на *2 * sizeof (float)* байт, а при відніманні зменшується на *3 * sizeof (float)* байт.

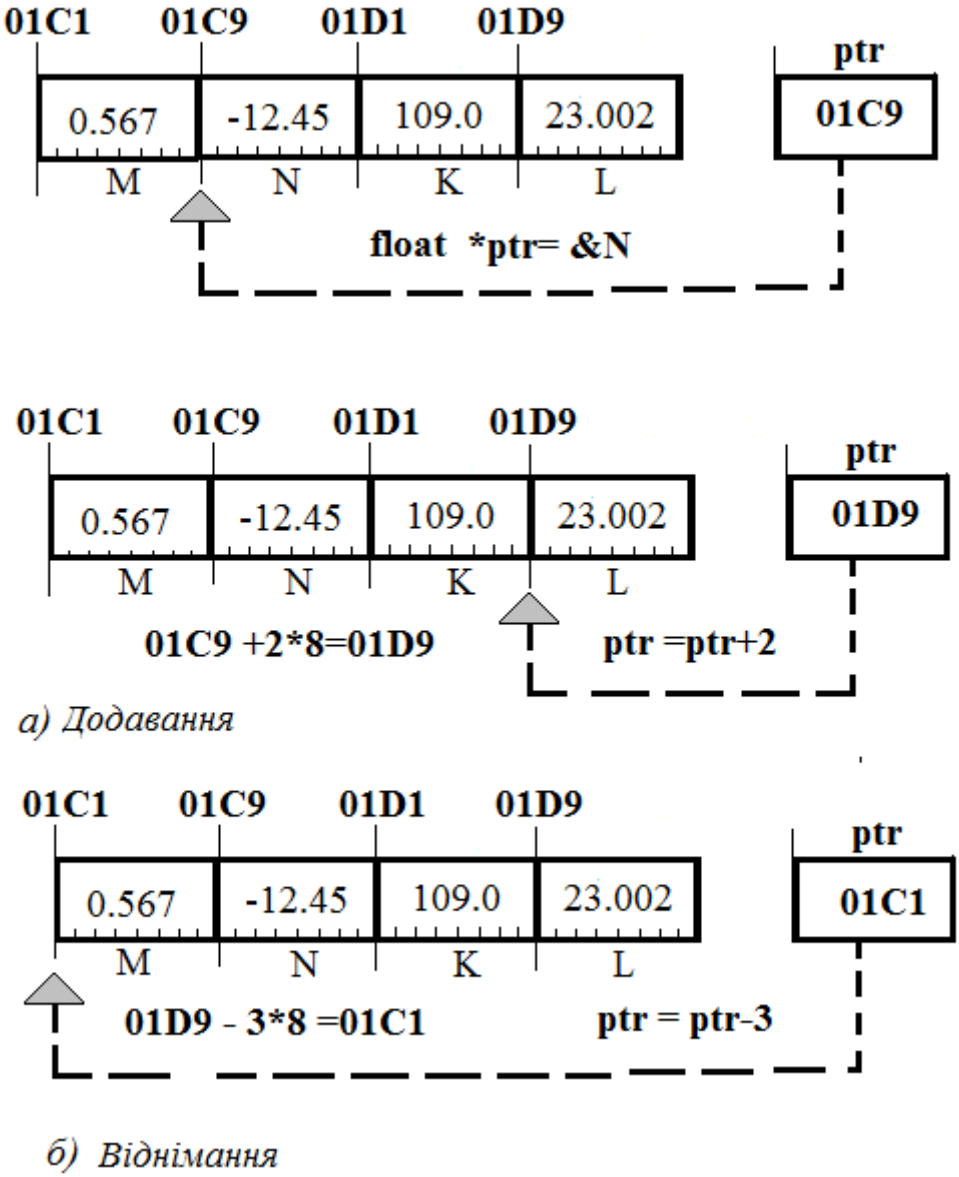

Рис 6. Умовна схема зміни покажчика при додаванні цілого числа *а* і віднімання з покажчика цілого числа *б*

Для збільшення покажчика на одиницю можна користуватися операцією інкремента *ptr* ++, а для зменшення на одиницю - операцією декремента *ptr* - -.

4.2.2 Операція віднімання двох покажчиків

Завдяки виконанню операції віднімання двох покажчиків можна визначити кількість об'єктів, розташованих між адресами, на які вказують ці покажчики; алеж, при цьому вважається, що тип об'єктів збігається з базовим типом покажчиків. Всі інші арифметичні операції заборонені. А саме: не можна

ділити і множити покажчики, підсумовувати два покажчика, виконувати над покажчиками побітові операції, підсумовувати покажчик зі значеннями, що мають тип *float* або *double* і т.п.

5. ПОКАЖЧИК VOID

Як відомо, при оголошенні змінної-покажчика необхідно вказувати його тип (наприклад, *int*, *float* і *char*). Вказівка типу дозволяє компілятору коректно виконати операції додавання і віднімання з покажчиком, умножаючи величину, яка додається або віднімається, на довжину типу для отримання зміщення. Однак, у деяких випадках цього робити не потрібно. Може виявитися, що в програмі досить просто отримати покажчик пам'яті, з яким вона буде працювати далі за призначенням.

У такому разі програма може створювати покажчик типу *void:*

*void *meinory_pointer;*

5.1 Безтиповий показчик

Замість типу даних при оголошенні покажчика можна поставити ключове слово *void*. Дане ключове слово означає, що цей покажчик був описаний «на що завгодно», тобто просто адреса в пам'яті. Будь-який покажчик автоматично приводиться до типу *void** – бестиповому покажчику *typeless pointer*. Інші покажчики, відповідно, називаються типізованим або тіпізованними *typed*. Приведення від *void* * до тіпізованного покажчика можливо за допомогою оператора явного приведення типу.

В мові С безтипові покажчики широко застосовуються для оперування шматками пам'яті або реалізації узагальнених функцій, які можуть працювати зі значеннями різних типів. В останньому випадку конкретний тип маскується за допомогою *void* («пустушка»).

В мові С існує особливий тип покажчиків – покажчики типу *void* або порожні покажчики. Ці покажчики використовуються в тому випадку, коли тип змінної невідомий. Так як *void* не має типу, то до нього не може бути застосована операція розадресації (взяття вмісту) і адресна арифметика, так як невідомо уявлення даних. Проте, якщо ми працюємо з покажчиком типу *void,* то нам доступні операції порівняння.

Таким чином, при знайомстві з покажчиками в C/C ++ може скластися спрощене уявлення, що покажчики можуть вказувати тільки на окремі змінні вбудованих типів C/C ++, і що це просто ще одна, альтернативна форма доступу до таких змінних. У такому застосуванні покажчики були б приємним доповненням мови, але з дуже обмеженими можливостями.

При більш уважному вивченні покажчиків C/C ++ виявляється, що покажчик може бути адресою розміщення (вказувати на) будь-якого допустимого об'єкта в програмі: структури, об'єкта класу, массива, функции, або знову ж покажчика на деякий об'єкт, або покажчика на покажчик і так далі… Це робить покажчики мало не найпотужнішим інструментом програміста на C ++ ... але і найнебезпечнішим в сенсі можливих прихованих помилок.

Контрольні питання

1. Дайте визначення покажчика.
2. Наведіть відомі Вам види покажчиків.
3. Яку операцію використовує програма для знаходження адреси? Наведіть приклад.
4. Який модифікатор використовується для виведення адреси змінної функцією *printf* ? Наведіть приклад.
5. Що таке ініціалізація покажчика?
6. Для чого призначена операція розіменування?
7. Який вигляд має операція розіменування покажчика *example*?
8. Які дві операції виконує оператор *float Z = * ptr*?
9. Які арифметичні операції допустимі над покажчиками? Що буде результатом цих операцій?
10. Які арифметичні операції заборонені над покажчиками?
11. В якому разі програма може створювати покажчик типу *void*? Наведіть приклад.

Література

1. Эффективный и современный C++. 42 рекомендации по использованию C++11 и C++14 [Текст] :учеб. пособие / Скотт Мейерс. – «Диалектика», 2019. – 304 с.
2. Qt4/7+ Практическое программирование на C++ [Текст] :учеб. пособие / Андрей Боровский. – БХВ-Петербург, 2012. – 496 с.
3. Технология программирования на C++. Начальный курс. [Текст]:учеб.пособие / Николай Литвиненко. – БХВ-Петербург, 2016. – 288 с.
4. Эффективное программирование на C++. Практическое программирование на примерах. [Текст] : учеб. пособие / Барбара Му, Эндрю Кенинг. – «Вильямс», 2016. – 368 с.
5. Язык программирования C++. Базовый курс. [Текст] :учеб. пособие / Стенли Липпман, Жози Лажойе, Барбара Му. – «Диалектика», 2016. – 1120 с.

Саксонов Геннадій Михайлович

Жукова Олена Андріївна

Сєчкін Ігор Арнольдович

ПОКАЖЧИКИ І МАСИВИ С \ С ++ У ПРИКЛАДАХ

Методичні вказівки до виконання лабораторних робіт з дисципліни «Програмування і алгоритмічні мови»

Видано в редакції авторів

Комп'ютерний дизайн, верстка та обробка – Г.М. Саксонов

Підписано до друку 16.03.2020. Формат 30х42/4.
Папір офсет. Ризографія. Ум. друк. арк. 0,9.
Обл.-вид. арк. 0,9. Тираж 5 пр. Зам. №

Національний технічний університет «Дніпровська політехніка»
49005, м. Дніпро, просп. Д. Яворницького, 19.