МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи
«Цикли з передумовою та постумовою у програмах мовою 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 № 3 від 28.12.09.

Харків НТУ «ХПІ» 2010

Методичні вказівки до лабораторної роботи «Цикли з передумовою та постумовою у програмах мовою 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. – Х. : НТУ «ХПІ», 2010. – 16 с.

Укладач М. І. Безменов

Рецензент Л. М. Любчик

Кафедра системного аналізу і управління

Мета роботи

Освоєння методики організації керування процесом обчислень за допомогою операторів циклу із передумовою і постумовою.

1. ТЕОРЕТИЧНІ ОСНОВИ

1.1. Загальні положення

Досить часто в програмі доводиться організовувати багаторазове повторення тих самих дій до виконання якої-небудь умови. Такі процеси називаються циклічними.

Загалом кажучи, існує два види циклів – цикл із передумовою і цикл з постумовою (див. рис. 1.1).

У першому випадку (рис. 1.1, *а*) спочатку проводиться перевірка деякої умови і, залежно від результату перевірки, або виконується, або пропускається сукупність операторів, що утворюють так називане тіло циклу. Якщо тіло циклу виконане, то процес повторюється, починаючи з перевірки умови. У другому ж випадку (рис. 1.1, *б*) спочатку виконується тіло циклу, після чого здійснюється перевірка умови повторення циклу. Якщо умова виконується, то процес повторюється.

1.2. Використання умовного оператора і оператора goto

Будь-який із циклів можна організувати з використанням оператора **`if`** та оператора переходу **`goto`**.

Будь-який виконуваний оператор може бути помічений. Мітка – це ідентифікатор, що міститься ліворуч від оператора і відокремлюється від нього двокрапкою. Наприклад,

```
M : y += x;
```

Усередині функції операторові, що має мітку, можна передавати керування за допомогою оператора переходу, що має вигляд:

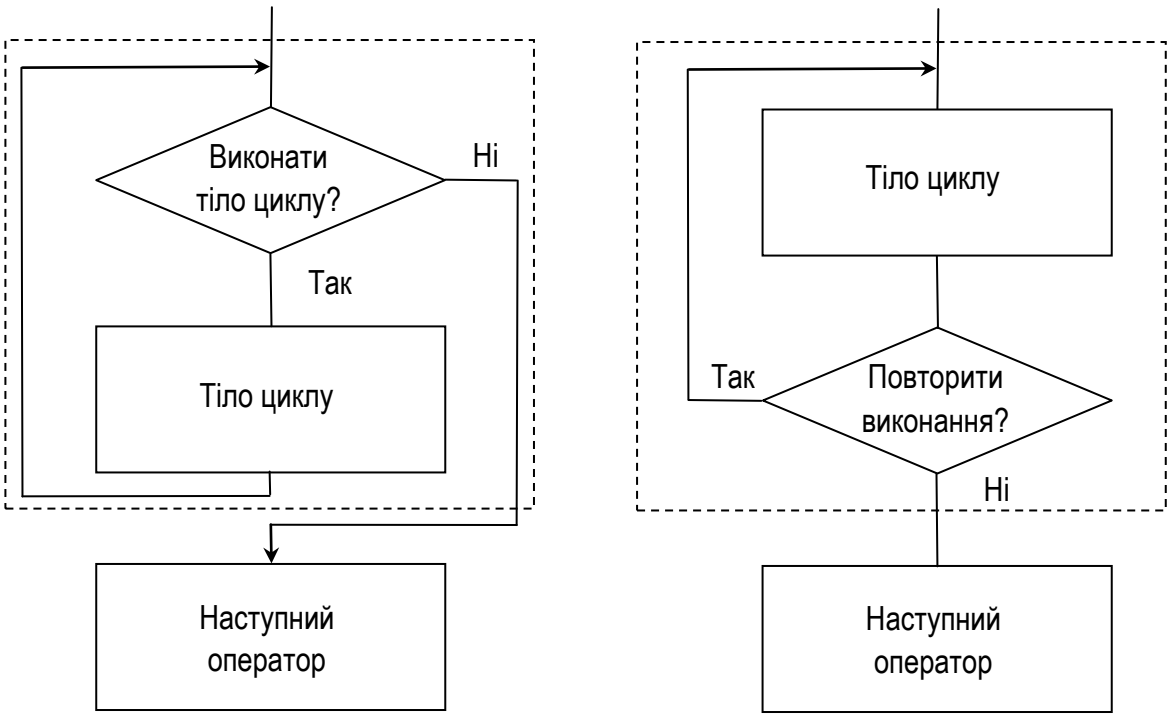
`goto` ідентифікатор`;`

де ідентифікатор – одна з міток.

Спеціально мітка ніяк не описується й відома усередині тільки тієї функції, у якій вона визначена. При цьому, ідентифікатор, що позначає мітку може збігатися з ідентифікатором, якого-небудь іншого об’єкта, тобто допустимою є, наприклад, навіть така послідовність операторів:

```
int M = 4;
M : M++;
```

Передавати керування можна усередину умовних операторів, перемикачів, операторів циклу, складених операторів, блоків, але робити це не рекомендується. Крім того, при передачі керування не можна обходити описи, що містять явну ініціалізацію.



а *б*

Рис. 1.1. Види циклів: *а* – цикл із передумовою; *б* – цикл з постумовою

Цикли можна організувати з використанням операторів, що перевіряють умову, і оператора переходу **goto**.

Організація циклу із передумовою за допомогою оператора **goto** може бути умовно зображена в такий спосіб:

```
мітка_1 : if (умова_припинення) goto мітка_2;
    тіло_циклу
    goto мітка_1;
мітка_2 : наступний_оператор
```

Дещо простіше виглядає формат циклу з постумовою:

```
мітка :
    тіло_циклу
```

```
if (умова_повторення) goto мітка;
наступний_оператор
```

Цикли, побудовані з використанням умовних операторів і операторів переходу, виглядають досить громіздко й не рекомендуються до використання у зв'язку з їхньою поганою структурованістю, тим більше, що сучасні вимоги до методики програмування констатують заборону використання оператора **goto** з деяким послабленням для виняткових випадків.

У мові C++ існують спеціальні оператори для організації циклів.

1.3. Оператор циклу із передумовою

Цей оператор у мові C++ задається конструкцією, що має такий вигляд:

```
while (вираз_умова) тіло_циклу
```

Оператор виконується в такий спосіб. Спочатку обчислюється значення виразу_умови і у випадку його істинності виконується тіло_циклу, після чого дії повторюються, починаючи з обчислення виразу_умови. Якщо вираз_умова виявиться помилковим, виконання циклу припиняється.

Особливості оператора **while**:

- як і в операторі **if**, виразом_умовою може бути як завгодно складний вираз, результатом обчислення якого є одне з булевих значень **true** або **false**, а також будь-який вираз, результат обчислення якого може бути автоматично перетворений або до значення **true**, або до значення **false**;
- якщо вираз_умова, записаний в заголовку циклу **while**, відразу ж має значення **false**, то цикл не виконається жодного разу;
- тілом_циклу може бути тільки один оператор (простий або складений) або ж блок;
- повинне бути забезпечене змінення хоча б однієї змінної, що входить у вираз_умову, причому це змінення, зрештою, повинне привести до прийняття виразом_умовою значення **false** (звичайно подібна зміна забезпечується в тілі_циклу, хоча у мові C++ це можна зробити і у самому виразі_умові);
- при некоректному записі циклу можлива ситуація, коли його виконання буде нескінченним, що може відбутися або через неправильний запис виразу_умови, або через помилки, що приводять до неправильної зміни значень змінних, що входять у вираз_умову;

- можливий примусовий вихід із циклу, для чого звичайно використовується оператор **break** (рідше **goto**);
- у деяких випадках спеціально організують нескінченний цикл, вихід з якого здійснюють за допомогою оператора **break**;
- у тілі циклу може бути присутнім оператор **continue**, виконання якого забезпечує пропуск всіх операторів до кінця циклу, тобто здійснює перехід на перевірку умови виконання циклу (вихід із циклу оператор **continue** не здійснює);
- усередині оператора **while** може бути записаний будь-який оператор, у тому числі будь-який оператор циклу;
- у випадку вкладеності циклів оператори **break** і **continue** діють тільки в самому внутрішньому з утримуючих їх операторів циклу (за допомогою оператора **break** можна вийти із внутрішнього циклу тільки на попередній рівень вкладеності).

У виразі_умові слід уникати використання операції «кома», оскільки в цьому випадку воно найчастіше за все буде помилковим. Наприклад, якщо цикл повинен виконуватися доти, поки значення деякої змінної w не вийде за межі інтервалу [a, b], то його заголовок може виглядати так:

```
while (w >= a && w <= b)
```

або

```
while (!(w < a || w > b))
```

У той же час заголовок

```
while (w >= a, w <= b)
```

помилковий.

Відзначимо, що використовуваний для примусового виходу із циклу оператор **break** найчастіше записується усередині оператора **if** з деякою додатковою умовою виходу:

```
if (додатковий_вираз_умова) {
  // ...
  break;
}
```

1.4. Оператор циклу з постумовою

Цей оператор у мові C++ задається конструкцією виду

```
do тіло_циклу while (вираз_умова);
```

Тут спочатку виконується тіло_циклу, після чого обчислюється значення виразу_умови. Якщо воно дорівнює **true**, тіло циклу виконується повторно. Процес повторюється допоки значенням виразу_умови не стане **false**. Усе, що сказано вище про особливості циклу **while**, має відношення і до циклу **do**. Єдина відмінність полягає в тому, що на відміну від оператора **while**, тіло_циклу в операторі **do** хоча б один раз виконується в обов'язковому порядку.

2. ПРИКЛАДИ ПРОГРАМ

Приклад 1. Дано додатне число ε. Обчислити

$$1+\frac{1}{3}+\frac{1}{2^2+2}+\frac{1}{2^3+3}+\cdots,$$

обмеживши тільки тими доданками, які перевищують ε.

```
#include <iostream.h>

int main()
{
   double Sum, eps, DegreeOfTwo;
   int n;

   cout << "eps = ";
   cin >> eps;
   cin.get();        // У потоці залишався символ кінця рядка
   n = 0;                 // Відлічуваний від 0 номер доданка
   Sum = 0;                          // Початкове значення суми
   DegreeOfTwo = 1;                        // Це 2 у степені 0
   while (1 / (DegreeOfTwo + n) > eps)
   {
     Sum += 1 / (DegreeOfTwo + n);
     DegreeOfTwo *= 2;           // Наступний степінь числа 2
     n++;
   }
   cout << "The sum is " << Sum << endl;
   cout << "Press <Enter>"; cin.get();
   return 0;
}
```

Приклад 2. Члени нескінченної послідовності дійсних чисел a_0, a_1, a_2, … задаються за формулою

$$a_i = \frac{(-1)^i}{2^i + i},\ \ i = 0, 1, 2, \ldots$$

Дано додатне число ε. Обчислити суму $a_0 + a_1 + \ldots + a_n$, де a_n – член послідовності, для якого вперше виконане співвідношення $|a_n| - |a_{n+1}| < \varepsilon$.

```
#include <iostream.h>
#include <math.h>
int main()
{
   double aNew, aOld, Sum, eps, DegreeOfTwo;
   int i, k;
   cout << "eps = "; cin >> eps;
   cin.get();        // У потоці залишався символ кінця рядка
   i = 0;                 // Відлічуваний від 0 номер доданка
   k = 1;                        // Нульовий степінь числа -1
   DegreeOfTwo = 1;                      // Це 2 у степені 0
   aNew = 1;                 // Значення доданка з номером 0
   Sum = 0;                        // Початкове значення суми
   do
   {
      aOld = aNew;
      Sum += aOld;
      i++;
      k = -k;                    // Наступний степінь числа -1
      DegreeOfTwo *= 2;                  // Це 2 у степені i
      aNew = k / (DegreeOfTwo + i);          // Новий доданок
   } while (fabs(aOld) - fabs(aNew) >= eps);
   cout << "The sum is " << Sum << endl;
   cout << "Press <Enter>";
   cin.get();
   return 0;
}
```

Приклад 3. Дано натуральне число *N*. З'ясувати, чи входить цифра 1 у запис числа N^2.

```
#include <iostream.h>
int main()
{
   int N2;                           // Другий степінь числа N
   bool Flag;

   cout << "N = ";
```

```
   cin >> N2;
   N2 *= N2;
   cin.get();        // У потоці залишався символ кінця рядка
   Flag = false;          // Вважаємо що 1 не входить у запис
   do
   {
      if (N2 % 10 == 1)            // Перевірка приналежності
      {
        Flag = true;                           // Так, входить
        break;                    // Примусовий вихід із циклу
      }
      N2 /= 10;                   // Знищення останньої цифри
   } while (N2 > 0);
   if (Flag)
     cout << "One enters into record of number\n";
   else
     cout << "One does not enter into record of number\n";
   cout << "Press <Enter>";
   cin.get();
}
```

Приклад 4. Здійснити генерування випадкових дійсних чисел у напіввідкритому інтервалу [0, 1), задавши як ознаку закінчення генерування появу числа, віддаоеного від середнього арифметичного раніш згенерованих чисел менш, ніж на 10 %. Скільки чисел було згенеровано? Визначити також мінімальне зі згенерованих чисел. Останнє згенероване число (ознаку закінчення генерування) не враховувати.

```
#include <iostream.h>
#include <math.h>
int main()
{
   double a;
   double Sum, Min;
   int n;
   randomize();
                                      // Генеруємо перше число
   Min = random(1000000) / 1000000.0;
   n = 1;                          // Кількість чисел дорівнює 1
   Sum  = Min;                        // Початкове значення суми
                                    // Генеруємо наступне число
   a = random(1000000) / 1000000.0;
   while (fabs(a - Sum / n) >= 0.001 * Sum / n)
```

```
   {
      if (a < Min) Min = a;                    // Новий мінімум
      n++;                                  // Нова кількість чисел
      Sum +=  a;                               // Нова сума чисел
      a = random(1000000) / 1000000.0;      // Наступне число
   }
   cout << "Quantity of numbers equally to " << n << endl;
   cout << "The minimal number is equal to " << Min << endl;
   cout << "Press <Enter>";
   cin.get();
}
```

Приклад 5. Дано натуральне число *n*. Знайти мінімальне натуральне число *m*, добуток цифр якого дорівнює *n*. Якщо таке число відсутнє, результатом повинне бути число 0. Для однозначного числа вважати добутком цифр саме це число.

```
#include <iostream.h>
#include <math.h>
int main()
{
   int n;
   int m, p, q;
   cout << "n = ";
   cin >> n;
   cin.get();
   m = 0;
   p = 0;
   while (p < n) {
     m++;
     q = m;
     p = 1;
     do {
        p = p * (q % 10);
        q = q / 10;
     } while (q > 0);
   }
   if (p > n) m = 0;
   cout << "m = " << m;
   cin.get();
   return 0;
}
```

Приклад 6. Дано дійсне число x ($x \neq 0$). Обчислити

$$\cfrac{x}{x+\cfrac{2}{x+\cfrac{4}{x+\cfrac{8}{x+\ddots\,\cfrac{256}{x}}}}}.$$

```
#include <iostream.h>
#include <math.h>
int main()
{
   float x, y;
   int n;
   int m, p, q;
   cout << "x = ";
   cin >> x;
   cin.get();
   n = 256;
   y = 0;
   while (n > 0) {
     y = n / (x + y);
     n = n / 2;
   }
   cout << "y = " << x * y;
   cin.get();
   return 0;
}
```

3. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

За час виконання лабораторної роботи, студент повинен:

1. Розробити алгоритм розв’язання задачі, запропонованої для програмування.
2. Здійснити програмну реалізацію розробленого алгоритму.
3. Здійснити відлагодження програми, виправивши синтаксичні та логічні помилки.
4. Підібрати тестові дані для перевірки програми, включаючи виняткові випадки.
5. Оформити звіт до лабораторної роботи.

6. Відповісти на контрольні запитання.
7. Здати викладачу працездатну програму з демонстрацією її роботи на декількох варіантах вихідних даних.

4. ВАРІАНТИ ЗАДАЧ

1. Дано додатні дійсні *x*, ε. У нескінченній послідовності $y_1, y_2, \ldots$, утвореній за законом

$$y_0 = x;\ y_i = \frac{1}{2}\left(y_{i-1} + \frac{x}{y_{i-1}} \right),\ i = 1, 2, \ldots,$$

знайти перший член y_n, для якого виконана нерівність $| y_n^2 - y_{n-1}^2 | < \varepsilon$.

2. Дано дійсні додатні числа *a*, ε. Послідовність $x_0, x_1, x_2, \ldots$ утворена за законом

$$x_0 = \begin{cases} \min(2a, \dfrac{6}{7}) & \text{при } a \le 1, \\ \dfrac{a}{7} & \text{при } 1 < a < 49, \\ \dfrac{a}{49} & \text{в інших випадках;} \end{cases}$$

$$x_k = \frac{5}{7} x_{k-1} + \frac{a}{6x_{k-1}^3},\quad k = 1, 2, \ldots$$

Знайти перший член x_k, для якого $\frac{7}{5} | x_{k+1} - x_k | < \varepsilon$. Обчислити для знайденого значення різницю $a - x_k^3$.

3. Дано ціле число $h > 1$. Одержати найбільше ціле *k*, для якого $3^k < h$.
4. Дано дійсні числа *x*, *y* ($x > 1$). Одержати найменше число виду x^n, що перевищує *y*, де *n* – ціле невід'ємне число.
5. Обчислити нескінченну суму

$$\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i(i+1)(i+2)}$$

із заданою точністю ε ($\varepsilon > 0$), вважаючи, що необхідна точність досягнута, якщо черговий доданок виявився за модулем менше, ніж ε.

6. Знайти корінь рівняння $y = ax^2 \sin x + b$ за умови, що корінь єдиний і перебуває на інтервалі [x_1, x_2]. Для знаходження кореня скористатися мето-

дом поділу відрізка навпіл. Точність обчислення кореня задається величиною $\varepsilon \geq 0$.

7. Дано ціле число. Поміняти порядок його цифр на зворотний.
8. Дано ціле число. Одержати нове число, видаливши в записі вихідного числа всі одиниці.
9. Реалізувати алгоритм Евкліда знаходження найбільшого спільного дільника (НСД) двох невід'ємних цілих чисел, який можна описати так. Нехай m і n – одночасно не рівні нулю цілі невід'ємні числа і нехай $m \geq n$. Тоді, якщо $n = 0$, то $\text{НСД}(m, 0) = m$, а якщо $n \neq 0$, то для чисел m, n і r, де r – остача від ділення m на n, виконується рівність $\text{НСД}(m, n) = \text{НСД}(n, r)$. Так, $\text{НСД}(18, 8) = \text{НСД}(8, 2) = \text{НСД}(2, 0) = 2$.

   Дано натуральні числа m, n. Використовуючи алгоритм Евкліда, знайти найбільший спільний дільник m і n.
10. Дано два натуральні числа. Знайти їх найменше спільне кратне.
11. Знайти натуральне число n, що подається двома різними способами сумою кубів двох натуральних чисел: $n = x^3 + y^3$ ($x \leq y$).
12. Дано дійсні числа x, ε ($\varepsilon > 0$). Знайти таке значення n, для якого вперше поточний доданок нескінченної суми

$$1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \ldots$$

   за модулем виявиться меншим ε. Переконатися, що наведена вище формула при досить малому значенні ε є поданням значення e^x.
13. Знайти «машинний нуль» в околі числа 1, тобто найбільше значення δ виду $\delta = 1/2^n$, для якого виконується рівність $1 + \delta = 1$.
14. Дано дійсне число x та натуральне число n. Обчислити

$$\cfrac{x}{1 + \cfrac{x}{3 + \cfrac{x}{5 + \cfrac{x}{\ddots \cfrac{x}{n + \cfrac{x}{n}}}}}}.$$

15. Дано дійсні числа a, b ($a < b$). Вводяться дійсні числа до першого числа, що не попадає в інтервал $[a, b]$. Знайти суму членів послідовності, що вводилися між двома останніми від'ємними значеннями. Якщо два

останніх від'ємних значення вводилися підряд або їхня кількість менше двох, дорівняти суму нулю.

16. Дано дійсне число ε ($\varepsilon > 0$). Знайти таке значення *n*, для якого вперше поточний доданок приведених нижче нескінченних сум за модулем виявиться меншим ε. Переконатися, що при досить малому значенні ε справедливі такі рівності:

$$\sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{(2k-1)!} = \sin 1.$$

17. Дано дійсне число ε ($\varepsilon > 0$). Знайти таке значення *n*, для якого вперше поточний доданок приведених нижче нескінченних сум за модулем виявиться меншим ε. Переконатися, що при досить малому значенні ε справедливі такі рівності:

$$\sum_{k=1}^{\infty} \frac{1}{k(k+1)\ldots(k+m)} = \frac{1}{m\,m!} \quad (m = 1,2,\ldots).$$

5. КОНТРОЛЬНІ ЗАПИТАННЯ

1. У чому особливість циклів із передумовою і постумовою?
2. Які оператори застосовуються для організації циклів?
3. Чим відрізняються оператори **while** і **do**?
4. Як здійснювати вибір між операторами **while** і **do**?
5. Чи є серед наведених нижче операторів потенційно нескінченні цикли? При позитивній відповіді вказати такі цикли.

```
while (true) {
  // Оператори
}

while (false) {
  // Оператори
}

do
  // Оператори
} while (true);

do
  // Оператори
} while (false);
```

6. Чи є серед наведених у попередньому запитанні операторів невиконувані цикли? При позитивній відповіді вказати такі цикли.
7. Чи можна яким-небудь способом завершити нескінченний цикл?
8. Знайдіть помилку в наведеному нижче фрагменті програми:

```
int sum = 0, counter = 0;
while (counter < 100 )
  sum += counter;
```

9. Знайдіть помилку в наведеному нижче фрагменті програми:

```
int sum = 0, counter = 50;
while (counter < 10) {
  sum += counter;
  counter--;
}
```

10. Знайдіть помилку в наведеному нижче фрагменті програми:

```
int sum, counter = 11;
do {
  sum = 0;
  sum += counter;
  counter++;
} while (counter != 40);
```

СПИСОК ЛІТЕРАТУРИ

1. Страуструп, Б. Язык программирования Си++ : Второе издание / Б. Страуструп. – К. : ДиаСофт, 1993. – Ч. 1. – 264 с. ; Ч. 2. – 296 с.
2. Керниган, Б. Язык программирования Си / Б. Керниган, Д. Ритчи. – М. : Финансы и статистика, 1992. – 272 с.
3. Либерти, Джесс. Освой самостоятельно С++ за 21 день : учеб. пособ. / Джесс Либерти. – М. : Издательский дом «Вильямс», 2001. – 816 с.
4. Подбельский, В. В. Программирование на языке Си / В. В. Подбельский, С. С. Фомин. – М. : Финансы и статистика, 1999. – 600 с.
5. Подбельский, В. В. Язык Си++ / В. В. Подбельский. – М. : Финансы и статистика, 1999. – 560 с.
6. Савитч, Уолтер. Язык С++. Курс объектно-ориентированного программирования / Уолтер Савитч. – М. : Издательский дом «Вильямс», 2001. – 704 с.

Навчальне видання

Методичні вказівки
до лабораторної роботи
«Цикли з передумовою та постумовою у програмах мовою С++»
з курсу «Програмування» для студентів напряму 6.040302 – Інформатика
і курсу «Програмування та алгоритмічні мови» для студентів напряму
6.040303 – Системний аналіз

Укладач БЕЗМЕНОВ Микола Іванович

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

За авторською редакцією

План 2010 р., поз. 9 / 39-10

Підп. до друку 15.03.2010 р. Формат 60 × 84 $^{1}/_{16}$. Папір офісний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 0,9. Наклад 50 прим.
Зам. № 60. Ціна договірна.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 3657 від 24.12.2009 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ «ХПІ», 61002, Харків, вул. Фрунзе, 21