МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

# МЕТОДИЧНІ ВКАЗІВКИ
## до лабораторної роботи
## «Обробка бінарних файлів з використанням стандартних засобів мови Сі»

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 «Обробка бінарних файлів з використанням стандартних засобів мови Сі» 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. – 16 с.

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

Рецензент І. П. Гамаюн

Кафедра системного аналізу і управління

# ВСТУП

На відміну від текстових файлів, де дані зберігаються у вигляді рядків різної довжини, бінарні файли характерні тим, що вони сприймаються як послідовність байт, яку можна довільно інтерпретувати в залежності від потреби. У такі файли можна записувати дані будь яких типів, але всі дані повинні мати той самий тип або в кожний момент, коли здійснюється читання елементу даних з файлу, треба знати, у якому типі цей елемент даних записувався у файл. Якщо у файл записувалися дані одного типу, то легко реалізується прямий доступ до кожного елементу даних.

**Метою** даної лабораторної роботи є освоєння методики використання бінарних файлів у програмах, написаних мовою Сі.

# 1. ТЕОРЕТИЧНІ ОСНОВИ

Зовнішнє ім’я файлу зазначається з урахуванням методики його подання для операційної системи. Воно може бути подане в повному вигляді (диск, шлях, ім’я: `"C:\\Users\\a.dat"`) або в скороченому – з урахуванням принципів умовчання, що закладені в роботу операційної системи, а саме:

- якщо не зазначене ім’я диска, то розглядається активний диск;
- якщо не зазначений шлях до файлу, то розглядається поточна папка (поточний каталог);
- якщо шлях починається не із символу `\`, то відлік шляху починається з поточної папки (слід пам’ятати про те, що в мові Сі символ `\` є управляючим і в разі необхідності його подання він подвоюється: `'\\'`).

## 1.1. Потокове введення/виведення

На рівні потокового введення/виведення обмін даними здійснюється за байтами. Такі введення/виведення можливі для друкувального пристрою, дисплея і дискових файлів. Функції бібліотеки введення/виведення, дозволяючи обробляти дані різних форматів, забезпечують при цьому буферизацію процесів введення і виведення.

Потік – це файл разом із засобами буферизації. При роботі з потоком можна:

- відкривати і закривати потоки (зв’язувати покажчики на потоки з конкретними файлами);
- вводити і виводити дані;
- аналізувати помилки потокового введення/виведення і умову досягнення кінця потоку (файлу);
- керувати буферизацією;
- керувати покажчиком поточної позиції у потоці.

Якщо потік відкритий, то з ним зв’язується так званий покажчик (індикатор) поточної позиції. Читання/записування даних здійснюється починаючи з того байту файлу, на який вказує цей покажчик. При цьому покажчик поточної позиції автоматично «переміщається» по файлу на ту кількість байт, що відповідає зчитаному/записаному обсягу інформації. Результатом є те, що наступний сеанс читання/записування здійснюватиметься з орієнтацією на нове положення покажчика поточної позиції.

## 1.2. Обробка бінарних файлів

Бібліотечні функції уведення/виведення підключаються через заголовний файл `stdio.h`, який, у свою чергу, автоматично підключається при підключенні файлу `iostream`.

Взагалі кажучи, файли зберігаються в бінарному (двійковому) форматі і тільки режим відкриття визначає, як програма повинна інтерпретувати вміст файлу. Текстовий режим є режимом за умовчанням, і в цьому режимі окремо інтерпретується послідовність з двох символів – «повернення каретки» (`'\xD'`, десятковий код 13) і «переведення рядка» (`'\xA'`, десятковий код 10). Бінарний режим відкриття файлу забезпечує сприйняття всіх символів окремо, у тому числі й символів `'\xD'` і `'\xA'`. Бінарні файли не створюються текстовими редакторами, але можуть бути створені за допомогою засобів мови C++ (або C).

Для забезпечення доступу до файлового потоку, як і при обробці текстових файлів, використовується покажчик на тип `FILE`, що визначений у заголовному файлі `stdio.h`:

```
FILE *ім’я_покажчика;
```

Перед початком роботи з потоком (як і для текстового файлу), його потрібно відкрити, для чого застосовується функція `fopen()`, яка зв’язує

потік з конкретним файлом, повертаючи при цьому значення покажчика на потік:

```
FILE *fopen(char *ім'я_файлу, char *режим_відкриття);
```

Параметри ім'я_файлу і режим_відкриття задаються або звичайними символьними масивами, або покажчиками на початок області пам'яті, у якій зберігається рядок символів.

Щоб вказати про необхідність відкриття файлу в бінарному режимі рядок, у якому вказується режим відкриття доповнюється літерою `b` відносно відповідного стандартного режиму для текстового файлу. Таким чином, режимами відкриття для бінарних файлів є:

- `"wb"` – відкриття нового бінарного файлу для записування;
- `"rb"` – відкриття існуючого бінарного файлу для читання;
- `"ab"` – бінарний файл відкривається (або створюється, якщо він відсутній) для дозаписування в кінець;
- `"wb+"` – новий бінарний файл відкривається для записування, читання і подальшого багаторазового виправлення з можливістю дозаписування в кінець файлу;
- `"rb+"` – існуючий бінарний файл відкривається як для читання, так і записування;
- `"ab+"` – бінарний файл відкривається або створюється (якщо він відсутній) як для читання, так і записування, причому записування здійснюється завжди в кінець файлу незалежно від положення покажчика поточної позиції.

Як і для текстових файлів, відкриття існуючого бінарного файлу в режимі `"wb"` або `"wb+"` забезпечує його повне відновлення.

Приклади відкриття бінарних файлів:

```
#include <iostream>
    // Текст програми
FILE *f1, *f2, *f3;
    // Текст програми
f1 = fopen("a.dat", "rb");
char OutFileName[] = "C:\\USERS\\result.dbl";
f2 = fopen(OutFileName, "wb");
char *FileName = new char [261];
cout << "\nInput the name of new file\n";
```

```
cin.getline(FileName, 261);
f3 = fopen (FileName, "w");
```

При помилках у відкритті потоку покажчик набуває значення NULL. Тому необхідно здійснювати перевірку значення покажчика, що можна зробити, наприклад, так:

```
FILE *f;
if ((f = fopen("a.dat", "rb")) == NULL)
{         // Дії при помилці
}
```

Закриття потоку здійснюється функцією

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

Повторне відкриття потоку потрібно передувати його закриттям.

При відкритті файлу в режимах "wb", "wb+", "rb", "rb+" покажчик поточної позиції установлюється на початок файлу, а при відкритті в режимах "ab" і "ab+" – на його кінець.

Для введення/виведення даних при роботі з бінарними файлами використовують такі функції:

size_t fread(**void** *покажчик_на_область_пам'яті, size_t розмір_блоку, size_t кількість, FILE *покажчик_на_потік); – читає з файлу, зв'язаного з потоком, на який указує покажчик_на_потік, кількість блоків даних, кожен з яких має розмір_блоку байт, і записує прочитані дані в область пам'яті, що адресується покажчиком, заданим як перший параметр; функція повертає кількість прочитаних блоків при успішному читанні або 0 у випадку, коли одразу виявляється кінець файлу або виникає помилка читання;

size_t fwrite(**void** *покажчик_на_область_пам'яті, size_t розмір_блоку, size_t кількість, FILE *покажчик_на_потік); – пише у файл, зв'язаний з потоком, на який указує покажчик_на_потік, кількість блоків даних, кожен з яких має розмір_блоку байт, читаючи дані з області пам'яті, що адресується покажчиком, заданим як перший параметр; функція повертає кількість записаних блоків при успішному записуванні або 0 у випадку помилки записування.

Тип size_t є цілочисловим типом, що в даному випадку еквівалентний типу **long**.

Для позиціонування в потоці можна переміщати покажчик поточної позиції за допомогою наступної функції:

```
int fseek(FILE * покажчик_на_потік, long зміщення,
          int початок_відліку);
```

При звертанні до цієї функції параметр зміщення задається змінною або виразом типу **long** і вказує, на скільки байт потрібно змістити покажчик поточної позиції у прямому (> 0) або зворотному (< 0) напрямку. Початок_відліку вказує, по відношенню до якої позиції здійснюється зсув покажчика поточної позиції. Він задається однією з трьох констант:

`SEEK_SET` ( `==  0`) – початок файлу;

`SEEK_CUR` ( `==  1`) – поточна позиція;

`SEEK_END` ( `==  2`) – кінець файлу.

Слід пам'ятати, що константа типу **long** записується у вигляді десяткового значення, слідом за яким додається суфікс `L` або `l` (наприклад, `128L`).

Функція `fseek()` повертає значення 0, якщо переміщення виконано успішно; інакше вона повертає нульове значення:

`fseek(f, -1L, SEEK_CUR);` – повернутися на 1 байт;

`fseek(f, 0L, SEEK_END);` – перейти на кінець потоку.

Деякі інші корисні функції:

**long** `ftell(FILE * покажчик_на_потік);` – повертає положення покажчика поточної позиції, що вимірюється в байтах від початку файлу, або значення (`-1L`) при помилці;

**int** `fgetpos(FILE * покажчик_на_потік, fpos_t * позиція);` – записує в область пам'яті, що адресується покажчиком позиція, положення покажчика поточної позиції файлу і повертає значення 0 при успішному виконанні або ненульове значення при помилці (тип `fpos_t` – це цілочисловий тип, що використовується для визначення положення покажчика поточної позиції файлу за допомогою функції `fgetpos()` або зміщення покажчика поточної позиції файлу функцією `fsetpos()`; див. нижче);

**int** `fsetpos(FILE * покажчик_на_потік, fpos_t * позиція);` – переводить покажчик поточної позиції файлу в позицію, яка зберігається в області пам'яті, що адресується покажчиком позиція, і повертає значення 0 при успішному виконанні або ненульове значення при помилці;

**void** `rewind(FILE * покажчик_на_потік);` – переміщає покажчик поточної позиції на початок потоку;

**int** feof(FILE * покажчик_на_потік); – повертає **ненульове** значення, якщо перед звертанням до цієї функції було прочитано ознаку кінця файлу, і 0, якщо спроби читання за кінцем файлу не було.

Якщо файл відкритий і для введення, і для виведення (режими "r+" і "a+"), то в разі потреби переходу від введення до виведення (або навпаки) необхідно обов'язково виконати примусове зміщення покажчика поточної позиції файлу (як варіант – на 0 байт відносно поточної позиції).

При роботі з файлами (будь-якими, а не тільки текстовими) корисними можуть бути такі дві функції, оголошені в заголовному файлі io.h:

**int** remove(**const char** *шлях); – знищення файлу, ім'я якого задане як параметр (файл повинен бути закритим);

**int** rename(**const char** *старе_ім'я, **const char** * нове_ім'я); – перейменування файлу або каталогу (файл повинен бути закритим). Якщо в новому імені файлу вказати інший шлях відносно старого, файл буде перенесений. Каталог перенести неможливо.

## 2. ПРИКЛАДИ ПРОГРАМ

**Приклад 1.** У текстовому файлі data.txt записано цілі числа, розділені довільною кількістю будь-яких пробільних символів, з можливими порожніми рядками в будь-якому місці файлу. Переписати в новий файл res.sol у форматі чотирибайтових цілих чисел зі знаком тільки ті з чисел, які кратні 3.

**Розв'язок.**

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

using namespace std;
int main()
{
    FILE *fInp, *fOut;
    long int m;
    char c;
                    // Існуючий файл відкриваємо для читання
    fInp = fopen("data.txt", "r");
    if (fInp == NULL)
        cout << "Error of opening of a input-file" <<
                " data.txt\n";
    else
    {
```

```
            // Відкриваємо новий файл у бінарному режимі
        fOut = fopen("res.sol", "wb");
        if (fOut == NULL)
            cout << "Error of opening of a output-file" <<
                    "res.sol";
        else
        {               // Читаємо до кінця файлу по одному
                           // блоку розміром sizeof(long) і
                  // пишемо в область пам'яті,де міститься
            while(fscanf(fInp, "%d", &m) != EOF)// змінна m
                if (m % 3 == 0)
                // Пишемо у файл sizeof(m) байт з адреси &m
                    fwrite(&m, sizeof(m), 1, fOut);
            fclose(fOut);   // Закриття результуючого файлу
            fclose(fInp);        // Закриття вхідного файлу
            cout << "Copying is completed!\n";
        }
    }
    cout << "Press any key to exit.";
    getch();
    return 0;
}
```

**Приклад 2.** Компонентами файлу data.dbl є дані, записані у форматі типу double. Поміняти місцями ліву і праву половини цього файлу, не використовуючи допоміжний файл. Якщо у файлі непарна кількість чисел, то число, що розміщене в середині файлу, не переміщати.

**Розв'язок.**

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

using namespace std;
int main()
{
   FILE *fInpOut;
   long int pLeft,         // Позиція в лівій половині файлу
            pRight,       // Позиція в правій половині файлу
            count;               // Кількість чисел у файлі
   double dL, dR;                        // Допоміжні змінні
                  // Відкриваємо існуючий бінарний файл для
   fInpOut = fopen("data.dbl", "r+b");  // читання і запису
   if (fInpOut == NULL)
```

```
      cout << "Error of opening of a file data.dbl\n";
   else
   {              // Встановлюємо покажчик поточної позиції
      fseek(fInpOut, 0L, SEEK_END);      // на кінець файлу
                     // Обчислюємо кількість чисел у файлі
      count = ftell(fInpOut) / sizeof(double);
                      //Обчислюємо першу позицію покажчика
                // поточної позиції у правій половині файлу
      pRight = (count - count / 2) * sizeof(double);
         // Як значення параметра циклу - позиція покажчика
                  // поточної позиції у лівій половині файлу
      for (pLeft = 0; pLeft < (count / 2) * sizeof(double);
         pLeft += sizeof(double), pRight += sizeof(double))
      {       // У циклі по черзі здійснюється встановлення
            // покажчика поточної позиції файлу на потрібну
             // позицію в лівій і правій половинах файлу зі
         // зчитуванням по sizeof(double) байт у змінні dL,
      // dR. Надалі знову по черзі виконується встановлення
            // покажчика поточної позиції файлу на потрібну
              // позицію в його правій і лівій половинах із
         // записом у файл умісту змінних dL, dR відповідно
         fseek(fInpOut, pLeft, SEEK_SET);
         fread(&dL, sizeof(double), 1, fInpOut);
         fseek(fInpOut, pRight, SEEK_SET);
         fread(&dR, sizeof(double), 1, fInpOut);
         fseek(fInpOut, pRight, SEEK_SET);
         fwrite(&dL, sizeof(dL), 1, fInpOut);
         fseek(fInpOut, pLeft, SEEK_SET);
         fwrite(&dR, sizeof(dR), 1, fInpOut);
      }
      fclose(fInpOut);                     // Закриття файлу
      cout << "Transformation is veiled!\n";
   }
   cout << "Press any key to exit.";
   getch();
   return 0;
}
```

**Приклад 3.** Дано два файли, що містять числа у форматі double. У якому з цих файлів мінімальне значення перебуває ближче до його початку?

**Розв'язок.**

```
#include <iostream>
#include <conio.h>
using namespace std;
long int PosMinInFile(FILE * f);                    // Прототип

int main()
{
   FILE *f1, *f2;
   long int p1, p2;
   char Name1[256], Name2[256];
   cout << "Enter a name of the first file:\n";
   cin.getline(Name1, 256);
   f1 = fopen(Name1, "rb");  // Відкриваємо перший бінарний
   if (!f1)          // файл для читання та робимо перевірку
     cout << "Error of opening of a file " << Name1;
   else
   {
     cout << "Enter a name of the second file:\n";
     cin.getline(Name2, 256);
            // Відкриваємо другий бінарний файл для читання
     if (!(f2 = fopen(Name2, "rb"))) // та робимо перевірку
       cout << "Error of opening of a file " << Name2;
     else
     {
        if ((p1 = PosMinInFile(f1)) == -1)
          cout << "File " << Name1 << " is empty";
        else
        {
          if ((p2 = PosMinInFile(f2)) == -1)
            cout << "File " << Name2 << " is empty";
          else
            if (p1 < p2)
              cout << "In a file " << Name1;
            else
              if (p1 > p2)
                cout << "In a file " << Name2;
              else cout << "Positions are equal";
        }
        fclose(f1);
        fclose(f2);
      }
```

```
    }
    cout << "\nPress any key to exit.";
    getch();
    return 0;
}
                                        // Визначення функції
long int PosMinInFile(FILE * f)
{
    double Min, d;
    long int i = 0, Result = -1;
      // Читаємо по одному блоку у sizeof(double) байт, поки
    while (fread(&d, sizeof(double), 1, f))       // функція
    {                          // fread() не поверне значення 0
       if (Result == -1 || d < Min)
       {
          Min = d;
          Result = i;
       }
       i++;
    }
    return Result;
}
```

## 3. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

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

1. Розробити алгоритм розв’язання задачі, запропонованої для програмування.
2. Здійснити програмну реалізацію розробленого алгоритму.
3. Здійснити налаштування 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.

## 4. ВАРІАНТИ ЗАДАЧ

1. Дано файл цілих чисел. Записати в інший файл найбільше значення перших 10 компонентів, потім – наступних 10 компонентів і т. д. Якщо в останній групі менше 10 чисел, то розглядати неповну групу.
2. Дано бінарний файл, що містить цілі невід'ємні числа. Переписати в інший файл числа, що читаються однаково зліва направо і справа наліво (спочатку додатні, а потім усі інші).
3. Дано бінарний файл, що містить цілі невід'ємні числа. Записати в новий файл парні компоненти початкового файлу у зворотному порядку.
4. У бінарному файлі $f$ записано цілі числа. Переписати його вміст у два файли. У перший файл записати всі парні числа з непарними номерами, а в другий – решту чисел. Дані в нових файлах повинні йти в порядку, зворотному порядку їх розміщення в початковому файлі.
5. Дано бінарний файл, що містить дійсні числа. Без використання іншого файлу переписати його вміст таким чином, щоб спочатку йшли всі додатні числа, а потім усі недодатні без зміни взаємного розташування елементів усередині кожної із цих двох груп.
6. Дано бінарний файл із ненульовими дійсними числами, кількість яких кратна 2. Розглядаючи числа, що міститься у файлі, послідовними парами, поміняти місцями елементи пар, якщо вони мають різні знаки.
7. Дано бінарний файл, що містить дійсні числа. Упорядкувати його вміст у такому порядку: перше, останнє, друге, передостаннє і т. д.
8. Дано впорядкований за зростанням бінарний файл, що містить цілі числа. Розширити цей файл, включивши в нього нове число зі збереженням властивості впорядкованості.
9. Дано бінарний файл $f$, що містить цілі числа. Записати у бінарний файл $g$ усі прості числа, що входять у файл $f$. Числа у файлі $g$ повинні йти:
   а) за неубуванням;
   б) у порядку убування без повторень.
10. Дано бінарний файл $f$, що містить цілі числа. Одержати файл $g$, утворений з файлу $f$ виключенням повторних входжень того самого числа. При вилученні даних з файлу зберігати перше з однакових чисел.
11. Уводяться цілі числа до першого нуля (всі числа різні). Знайти, де сума елементів більше: до максимуму або після. Відповідний фрагмент

списку введених даних зберегти в бінарному файлі, записавши дані в порядку, зворотному порядку введення.

12. Дано бінарний файл цілих чисел, кількість яких кратна 4. Числа у файлі йдуть у такому порядку: два парні, два непарні, два парні, два непарні і т. д. Змінити вміст файлу таким чином, щоб порядок чисел був таким: парне, непарне, парне, непарне і т. д.
13. Дано натуральне число $k$ і бінарний файл, що містить цілі числа, кількість яких кратна $4k$. Числа у файлі йдуть послідовними групами з $k$ парних і $k$ непарних чисел. Поміняти місцями групи таким чином, щоб числа йшли послідовними групами:
    а) з $2k$ парних і $2k$ непарних значень;
    б) з $2k$ непарних і $2k$ парних значень.
14. Дано бінарний файл, що містить цілі числа. Після кожного непарного числа вставити цілу частину його половини.
15. Дано два бінарні файли $f_1$, $f_2$ з числовими даними одного й того самого типу. Сформувати файл $f_3$, що є «різницею» файлів $f_1$ і $f_2$, тобто містить усі числа з файлу $f_1$, які не входять у $f_2$.
16. Дано два бінарні файли $f_1$, $f_2$ з числовими даними одного й того самого типу. Сформувати файл $f_3$, який є перетином файлів $f_1$ і $f_2$, тобто містить без повторень усі числа, що одночасно входять як у $f_1$, так і в $f_2$.
17. Дано бінарний файл, що містить цілі числа. Вивести кількість ділянок цього файлу, що складаються:
    а) з послідовно розміщених парних елементів;
    б) з послідовно розміщених однакових елементів.
18. Дано бінарний файл, що містить числа відмінні від нуля. Компоненти цього файлу записані в наступному порядку: п’ять додатних, п’ять від’ємних, п’ять додатних, п’ять від’ємних і т.д., причому кількість чисел у файлі кратна 20. Переписати його вміст в інший файл так, щоб числа йшли в наступному порядку: десять від’ємних, десять додатних, десять від’ємних, десять додатних і т. д.
19. У бінарному файлі зберігаються відомості про результати сесії студентів (прізвища довжиною не більше 20 символів і по 5 оцінок). Упорядкувати вміст файлу за незростанням середніх балів.

## 5. КОНТРОЛЬНІ ЗАПИТАННЯ

1. Для чого використовуються файли даних?
2. Що таке потік і які дії можна виконувати над ним?
3. Яке призначення покажчика поточної позиції?
4. У чому особливість бінарних файлів по відношенню до текстових?
5. Які попередні дії треба виконати для забезпечення роботи з бінарним файлом?
6. Охарактеризуйте режими відкриття бінарних файлів. Як режими відкриття задаються при використанні засобів мови Сі?
7. Перелічіть і охарактеризуйте функції для введення/виведення даних при роботі з бінарними файлами.
8. До яких наслідків приводить читання ознаки кінця файлу під час уведення даних з файлу? Що рекомендується робити для запобігання цих наслідків?
9. За допомогою чого можна керувати положенням поточного покажчика?
10. Як здійснюється закриття потоку?
11. Як дізнатися положення поточного покажчика файлу?
12. У файлі зберігаються дані типу **`double`**. Як визначити кількість чисел, що містяться в цьому файлі?
13. Як можна повернутися на початок файлу, не здійснюючи повторне відкриття потоку?
14. У бінарний файл потрібно записати константу, що задана літералом (тобто явно; наприклад, числове значення `3.1415` або текстовий рядок `"Result\n"`). Як це можна зробити?
15. Чи можна записати в бінарний файл уміст масиву, не організуючі перебирання його елементів у циклі? При позитивній відповіді навести приклад виведення.
16. Як можна виконати перейменування файлу?
17. Яка функція може бути використана для знищення файлу?

## СПИСОК ЛІТЕРАТУРИ


1. Керниган, Б. Язык программирования Си / Б. Керниган, Д. Ритчи. – М. : Финансы и статистика, 1992. – 272 с.
2. Павловская, Т. А. C/C++. 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.

Навчальне видання

Методичні вказівки
до лабораторної роботи
«Обробка бінарних файлів з використанням
стандартних засобів мови Сі»
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. 104.

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. № 250. 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.