МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ
Національний технічний університет України
“Київський політехнічний інститут”

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ СОРТУВАННЯ ТА ДВІЙКОВОГО ПОШУКУ НА БАГАТОВИМІРНИХ МАСИВАХ

Методичні вказівки та завдання до виконання
курсової роботи по кредитному модулю
«Структури даних та алгоритми. Курсова робота»
для студентів денної форми навчання напрямку 6.050102
«Комп’ютерна інженерія»

Рекомендовано вченою радою факультету прикладної математики НТУУ «КПІ»

Київ 2012

МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ
Національний технічний університет України
“Київський політехнічний інститут”

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ СОРТУВАННЯ ТА ДВІЙКОВОГО ПОШУКУ НА БАГАТОВИМІРНИХ МАСИВАХ

Методичні вказівки та завдання до виконання
курсової роботи по кредитному модулю
«Структури даних та алгоритми. Курсова робота»
для студентів денної форми навчання напрямку 6.050102
«Комп’ютерна інженерія»

Затверджено
на засіданні кафедри системного
програмування і спеціалізованих
комп’ютерних систем
ФПМ НТУУ «КПІ»
Протокол № 6 від 12.12.2012

Київ НТУУ «КПІ» 2012

УДК 004.42

Дослідження ефективності алгоритмів сортування та двійкового пошуку на багатовимірних масивах: методичні вказівки та завдання до виконання курсової роботи з дисципліни «Структури даних та алгоритми» для студентів денної форми навчання напрямку підготовки 6.050102 «Комп'ютерна інженерія» [Електронне видання] / О.І.Марченко. – К.: НТУУ «КПІ», 2012. – 66 с.

Гриф надано вченою радою ФІПМ
(протокол № 5 від 24.12.2012 р.)

Навчальне електронне видання містить завдання та методичні вказівки до виконання курсової роботи по кредитному модулю «Структури даних та алгоритми. Курсова робота».

Наведені варіанти індивідуальних завдань, надаються методичні вказівки щодо виконання завдання, проведення дослідження, тестування програм та оформлення звіту. Призначені для студентів очної форми навчання напряму підготовки 6.050102 «Комп'ютерна інженерія».

Навчальне електронне видання

Автор: *Марченко Олександр Іванович*, канд.техн.наук, доц.

Відповідальний редактор: *Орлова М.М.,кандидат техн. наук,доцент.*
Рецензент: Заболотня *Т.М., канд. техн. наук, ст.викладач*

За редакцією автора

ЗМІСТ

ВСТУП

Дисципліна "Структури даних та алгоритми" є базовою спеціальноюнормативною дисципліною підготовки фахівців рівня бакалавр з напрямку 050102 «Комп'ютерна інженерія», а також студентів спеціальностей 8.091.501 "Комп'ютерні системи та мережі", 8.091.502 "Системне програмування" та 8.091.503 "Спеціалізовані комп'ютерні системи" і викладається у 1-му та 2-му семестрах.

Вона повинна передувати всім програмістським дисциплінам крім початкового курсу програмування, з яким викладається одночасно.

Кредитний модуль «Структури даних та алгоритми. Курсова робота» для фундаментального засвоєння теоретичних знань і практичних навичок, отриманих під час вивчення дисципліни «Структури даних та алгоритми».

Курсова робота складається з практичної частини, яка полягає у реалізації заданих алгоритмів для рішення задач на багатовімірних масивах з використанням принципів структурного та модульного програмування, а також з теоретичної дослідницької частини, яка полягає у вимірюванні часу роботи алгоритмів як для звичайних випадків одновимірного масиву, так і різних випадків поведінки алгоритмів на багатовимірних масивах, і у дослідженні причин неоднакової поведінки цих алгоритмів на одновимірних і багатовимірних масивах.

Курсова робота дозволяє отримати досвід реалізації і дослідження складних алгоритмів на складних структурах даних, а також практичного засвоєння принципів структурного і модульного програмування і оформлення документації на створений програмний продукт.

Методичні вказівки включають:

- постановку завдання та вимоги до виконання програми, що повинні розробити студенти;
- методичні вказівки до вимірювання часу роботи алгоритмів;
- вказівки до оформлення звіту та тестування розробленого алгоритму і відповідної йому програми;
- контрольні питання для самоконтролю та підготовки до захисту лабораторної роботи;
- варіанти індивідуальних завдань.

ТЕХНІЧНЕ ЗАВДАННЯ НА КУРСОВУ РОБОТУ

I. Описати принцип та схему роботи кожного із досліджуваних методів сортування або пошуку для одновимірного масиву.

II. Скласти алгоритми сортування або пошуку в багатовимірному масиві заданими методами, згідно до варіанту, та написати відповідну програму на мові програмування.

Програма повинна задовольняти наступним вимогам:

1. Всі алгоритми повинні бути реалізовані в рамках ОДНІЄЇ програми з діалоговим інтерфейсом для вибору варіантів тестування та виміру часу кожного алгоритму.

2. Одним з варіантів запуску програми має бути режим запуску виміру часу всіх алгоритмів у пакетному режимі, тобто запуск всіх алгоритмів для всіх випадків і побудова результуючої таблиці за наведеним нижче зразком для масиву з заданими геометричними розмірами.

3. При реалізації програми повинні бути використані модулі (unit).

4. Програма повинна мати коментарі для всіх структур даних, процедур та функцій, а також до основних смислових фрагментів алгоритмів.

III. Виконати налагодження та тестування коректності роботи написаної програми.

IV. Провести практичні дослідження швидкодії складених алгоритмів.

V. За результатами досліджень скласти порівняльні таблиці за різними ознаками.

Одна таблиця результатів сортування для масиву з заданими геометричними розмірами повинна мати такий вигляд:

Таблиця № для масиву A[p,m,n], де p= ; m= ; n= ;

	Впорядко-ваний	Невпорядко-ваний	Обернено впорядкований
Назва алгоритму 1			
Назва алгоритму 2			
Назва алгоритму 3			

Для варіантів курсової роботи, де порівнюються три способи обходу, замість назв алгоритмів записуються назви способів обходу.

Одна таблиця результатів двійкового пошуку для масиву з заданими геометричними розмірами повинна мати такий вигляд:

Таблиця № для масиву A[p,m,n], де p= ; m= ; n= ;

	після ¼ порів-нянь	після ½ порів-нянь	після ¾ порів-нянь	при остан-ньому порівня-нні	елемента немає в області пошуку
Двійковий пошук №1					
Двійковий пошук №2					

Для виконання ґрунтовного аналізу алгоритмів потрібно виконати виміри часу та побудувати таблиці для декількох масивів

з різними геометричними розмірами, а також для стандартного випадку одномірного масиву. Кількість необхідних таблиць для масивів з різними геометричними розмірами залежить від задачі конкретного варіанту курсової роботи і вибирається самостійно студентом так, щоб виконати всебічний та ґрунтовний порівняльний аналіз заданих алгоритмів. За отриманими результатами можуть бути також побудовані графіки для наочності подання інформації.

VI. Виконати порівняльний аналіз поведінки заданих алгоритмів за отриманими результатами:

- для одномірного масиву відносно загальновідомої теорії;
- для багатовимірних масивів відносно результатів для одномірного масиву;
- для заданих алгоритмів на багатовимірних масивах між собою;
- дослідити вплив різних геометричних розмірів багатовимірних масивів на поведінку алгоритмів та їх взаємовідношення між собою;
- для всіх вищезазначених пунктів порівняльного аналізу пояснити, ЧОМУ алгоритми в розглянутих ситуаціях поводять себе саме так, а не інакше.

VII. Зробити висновки за виконаним порівняльним аналізом.

VIII. Програму курсової роботи під час її захисту ОБОВЯЗКОВО мати при собі на електронному носії інформації.

СТРУКТУРА ЗВІТУ КУРСОВОЇ РОБОТИ

1. Перший титульний аркуш.
2. Другий титульний аркуш.
3. Аркуш технічного завдання.
4. Опис теоретичних положень.
5. Схема імпорту/експорту модулів та структурна схема взаємовикликів процедур та функцій.
6. Опис призначення процедур і функцій.
7. Текст програми з коментарями.
8. Тести.
9. Результати (таблиці виміру часу).
10. Порівняльний аналіз алгоритмів**!!!**
11. Висновки по отриманих результатах**!!!**
12. Список літератури.
13. Пронумерувати сторінки починаючи з другої.

Звіт курсової роботи у вигляді файлу формату текстового редактора *MS WORD-**2003***, а також текст програми курсової роботи на мові програмування разом з файлами даних повинні бути:

1) зархівовані у архів з ідентифікатором виду **KV–XX_PRIZVYSCHE** (ідентифікатор записувати **ТІЛЬКИ ЛАТИНСЬКИМИ** буквами);

2) вислані електронною поштою на адресу **sda_kr@ukr.net**.

ВИМОГИ ДО ВИКОНАННЯ КУРСОВОЇ РОБОТИ

Семестровий рейтинг з курсової роботи «Дослідження алгоритмів» дисципліни «Структури даних та алгоритми» складається з рейтингових балів (див. табл. 1) і не перевищує 100 балів. Якщо студент з урахуванням заохочувальних балів отримує більше 100 балів, йому проставляється 100 балів.

Таблиця 1. Система рейтингових балів.

№	Контрольний захід	Бали
	ПЕРЕВІРКА ЗВІТУ	
1	Загальний вигляд та правильність структури звіту.	2
2	Правильність оформлення титульних листів.	3
3	Правильність оформлення технічного завдання.	3
4	Якість викладення теоретичних положень.	5
5	Якість схеми взаємозв'язків модулів та схеми взаємовикликів процедур і функцій програми курсової	4
6	роботи.	4
7	Якість опису призначення процедур і функцій.	5
8	Використання модулів у програмах.	5
9	Всі алгоритми реалізовані правильно та однаково ефективно.	5
10	Коректність реалізації виміру швидкодії алгоритмів.	3
11	Якість коментарів у програмі.	3
12	Наявність достатньої кількості тестів.	5
13	Наявність достатньої кількості таблиць результатів виміру часу.	5
14	Правильність вибору розмірів структур даних (масивів) для отримання результатів, які дозволяють коректне порівняння результатів.	5
15	Наявність вимірів швидкодії та порівняння з вектором.	5
16	Наявність порівняльного аналізу алгоритмів на багатовимірних масивах.	16
17	Повнота, коректність та конкретність порівняльного аналізу результатів у висновках. Наявність переліку використаної літератури.	2
	Разом за звіт	**80**

	ЗАХИСТ	
18	Ступінь володіння матеріалом.	10
19	Аргументованість відповідей.	10
	РАЗОМ	**100**

Останній день здавання звітів курсової роботи на перевірку встановлюється на останньому або передостанньому тижні квітня.

Якщо звіт курсової роботи був зданий несвоєчасно, застосовуються **штрафні санкції** за схемою показаною у Таблиці 2.

Таблиця 2. Штрафні санкції за несвоєчасно зданий звіт.

№	Термін здавання звіту КР	Штрафна санкція
1	Курсова робота здана до 15 травня	Студент отримує оцінку не вище B, незалежно від набраних балів.
2	Курсова робота здана до 22 травня	Студент отримує оцінку не вище C, незалежно від набраних балів.
3	Курсова робота здана на тижні, коли починаються захисти курсових робіт у будь-якій групі, але перед захистом у своїй групі.	Студент отримує оцінку не вище D, незалежно від набраних балів.
4	Курсова робота здана після захисту в своїй групі	Студент отримує оцінку не вище E, незалежно від набраних балів.

Заохочувальні бали можуть бути нараховані за здавання курсової роботи до 20 квітня (5 балів), якщо якість зданого звіту заслуговує не менше 70 балів; за оригінальність та особливу докладність висновків (від 2-х до 10-ти балів); за виконання курсової роботи за індивідуальним завданням підвищеної складності (від 10 до 20 балів).

Заохочувальні бали за не нараховуються, якщо виявлена несамостійність виконання курсової роботи.

У випадку виявлення несамостійності виконання курсової роботи бали за захист курсової роботи студенту не нараховуються, а також нараховується до 20 **штрафних балів** в залежності від рівня несамостійності або відразу проставляється оцінка FX (незадовільно).

Студент допускається до захисту, якщо його рейтинг за звіт з урахуванням штрафних та заохочувальних балів складає не нижче, ніж 40 балів.

У випадку, якщо поточний рейтинг студента за другий семестр на початок тижня захистів курсових робіт складає не менше 75 балів (тобто A, B або C), то він має право, не захищаючи курсової роботи, отримати оцінку за курсову роботу “автоматом” відповідно до кількості балів за якість звіту плюс 20 балів за захист.

Таблиця 3. Відповідність між рейтингом та оцінкою за курсову роботу.

Рейтинг	Оцінка ECTS	Традиційна оцінка
95≤RD≤100	A – відмінно	Відмінно
85≤RD≤94	B – дуже добре	Добре
75≤RD≤84	C – добре	
65≤RD≤74	D – задовільно	Задовільно
60≤RD≤64	E – достатньо	
40≤RD≤59	FX – незадовільно	Незадовільно
0≤RD≤39	F – незадовільно (потрібна додаткова робота)	

МЕТОДИЧНІ ВКАЗІВКИ
З ВИМІРЮВАННЯ ЧАСУ РОБОТИ АЛГОРИТМІВ

Вимірювання часу в середовищі DOSBox

Середовище DOSBox моделює роботу персональних комп'ютерів з процесорами від Intel-286 до Intel-486 і підтримує коректну роботу програм, що працюють в операційній системі DOS. Частота процесора, що моделюється, встановлюється «гарячими клавішами»:

Ctrl+F12 – збільшення частоти;

Ctrl+F11 – зменшення частоти.

Структури даних для вимірювання часу

Для дослідження швидкодії алгоритмів в середовищі DOSBox рекомендується використовувати процедуру GetTime з модуля Dos мови Turbo Pascal, яка повертає поточне значення системного часу комп'ютера.

```
procedure GetTime (var години, хвилини, секунди,
соті_долі_секунди : word);
{ Значення параметрів знаходяться в діапазонах:
     "години":  0..23;
     "хвилини":  0..59;
     "секунди":  0..59;
     "соті_долі_секунди":  0..99.
}
```

Приклад вимірювання часу роботи алгоритму

```
program MYPROG;
uses Dos;

type
   TTime = record   {Тип для зберігання часу}
     Hours,           {години}
     Minutes,         {хвилини}
     Seconds,         {секунди}
     HSeconds: Word   {соті долі секунди}
    end;

var
   StartTime,FinishTime: TTime;{Час старту та фінішу
алгоритму}
   Algorithm1_Time : Longint; {Час роботи алгоритму}

function ResTime(const STime, FTime: TTime): Longint;
{Визначає різницю між часом закінчення і часом старту
алгоритму}

begin
   ResTime := 36000 * Longint(FTime.Hours) +
               6000 * Longint(FTime.Minutes) +
                100 * Longint(FTime.Seconds) +
                      Longint(FTime.HSeconds) -
              36000 * Longint(STime.Hours) -
               6000 * Longint(STime.Minutes) -
                100 * Longint(STime.Seconds) -
                      Longint(STime.HSeconds);
end;
            {інші описи алгоритму}

BEGIN
                {Початок алгоритму}

   with StartTime do
GetTime(Hours,Minutes,Seconds,HSeconds);

                {Досліджуваний фрагмент алгоритму}

   with FinishTime do
GetTime(Hours,Minutes,Seconds,HSeconds);
```

```
    Algorithm1_Time := ResTime(StartTime, FinishTime);
    writeln('Algorithm1 time = ', Algorithm1_Time)

                    {Закінчення алгоритму}
END.
```

Вимірювання часу в середовищі Windows

Структури даних для вимірювання часу

Для вимірювання часу роботи алгоритмів в середовищі Windows рекомендується використовувати:

тип **Int64 = array[0..3] of Word;**

масив **time_64: array[0..1] of Int64;**

константу **ptime: ^LongInt = @time_64[1];**

і процедури **rdtsc** і **sub_int64,**

приклад використання яких наведено нижче.

Врахування похибки

Оскільки операційна система Windows є багатозадачною і у ній, окрім прикладної програми, одночасно працює кілька системних процесів, то отриманий час має похибку.

Для врахування цієї похибки рекомендується виконувати вимірювання часу для кожного випадку 14 разів, перші два значення відкинути, після цього відкинути максимальне та мінімальне з 12 значень, що залишились, і в результуючу таблицю записати середнє арифметичне значення з решти.

Приклад вимірювання часу роботи алгоритму

```
program TimerExp;
uses dos;
const
    n=20;
type
    { Тип Int64 служить для зчитування значень
таймера у тактах }
    Int64 = array[0..3] of Word;

    mas = array[1..n] of word; { тип масиву, що
сортується }

var
    { масив time_64 містить 2 елементи для запису
значення часу
      до і після виконання алгоритму, а також для
запису різниці між ними }
      в елементі time_64[1] }
    time_64: array[0..1] of Int64;

    x:mas;          { масив, що сортується }
    i,j,k:integer;
    b:word;

const
    { Типізована константа-вказівник ptime служить
для зчитування результату заміру часу (різниці між
кінцевим та початковим значеннями таймера)
      із другого елемента масиву time_64[1]}
    ptime: ^LongInt = @time_64[1];


{ Процедура rdtsc зчитує поточне значення таймера в
тактах у параметр int_64 }
{ Оскільки ця процедура написана на асемблері, то
результат буде доступний у процедурі, де був
зроблений виклик. }


procedure rdtsc(int_64: Int64); assembler;
```

```
asm
   push   es
   db $0f,$31
   lesdi,int_64
db $66;   movdi],ax
db $66;   movdi][4],dx
   popes
end;

{ Процедура sub_int64 обчислює різницю між значеннями
параметрів op1 і op2, а також записує різницю у
параметр op1}

procedure sub_int64(op1, op2: Int64); assembler;
asm
   push   es
   push   ds
   lesdi,op1
   ldsbx,op2
db $66;   movax,[bx]
db $66;   subdi],ax
db $66;   movax,[bx][4]
db $66;   sbbdi][4],ax
   popds
   popes
end;

{-----------------------------}


procedure sort1(var a:mas);
var
    i:integer;
    b:Word;
begin

   rdtsc(time_64[0]);


   for i:=k+1 to n do
   begin
          b:=a[i];
          j:=i;
```

```
            while (j>1)and(a[j-1]>b) do
            begin
                 a[j]:=a[j-1];
                 j:=j-1;
            end;
            a[j]:=b;
    end;

    rdtsc(time_64[1]);
    sub_int64(time_64[1],time_64[0]);

    writeln('SortTime:',ptime^);

end;

begin
    randomize;

    for i:=1 to n do x[i]:=i;
    writeln('Time for sorted array:');
    sort1(x);

    for i:=1 to n do x[i]:=random(100);
    writeln('Time for random array::');
    sort1(x);

    for i:=1 to n do x[i]:=n-i+1;
    writeln('Time for reverse sorted array:');
    sort1(x);

    readln;
end.
```

ЗАДАЧІ

1. Впорядкувати окремо кожен переріз тривимірного масиву A [p,m,n] наскрізно по рядках за незменшенням.
2. Впорядкувати окремо кожен переріз тривимірного масиву A [p,m,n] наскрізно по стовпчиках за незменшенням.
3. Впорядкувати окремо кожен переріз тривимірного масиву A [p,m,n] таким чином: переставити стовпчики перерізу за незменшенням значень елементів його першого рядка.
4. Впорядкувати окремо кожен переріз тривимірного масиву A [p,m,n] таким чином: переставити стовпчики перерізу за незменшенням сум їх елементів.
5. Впорядкувати тривимірний масив A [p,m,n] таким чином: переставити перерізи масиву за незменшенням значень вектору перших елементів кожного перерізу A [*,1,1].
6. Впорядкувати тривимірний масив A [p,m,n] таким чином: переставити перерізи масиву за незменшенням сум їх елементів.
7. Впорядкувати головну та побічну діагоналі всіх перерізів тривимірного масиву A [p,n,n] за незменшенням (побічну діагональ у напрямку від правого верхнього кута). Тим фактом, що після сортування головної діагоналі під час сортування побічної діагоналі центральний елемент може бути замінений на інший, порушивши відсортованість головної діагоналі, знехтувати.

8. Визначити знаходження та місцеположення заданого елемента X серед елементів діагоналей всіх перерізів тривимірного масиву A[p,n,n]. Елементи кожного перерізу окремо впорядковані наскрізно по рядках за незбільшенням.
9. Визначити знаходження та місцеположення заданого елемента X серед елементів діагоналей всіх перерізів тривимірного масиву A[p,n,n]. Елементи кожного перерізу окремо впорядковані наскрізно по стовпчиках за незменшенням.
10. Визначити знаходження заданого елемента X серед всіх елементів тривимірного масиву A[p,m,n]. Елементи кожного рядка окремо у кожному перерізі впорядковані за незбільшенням.
11. Визначити знаходження заданого елемента X серед всіх елементів тривимірного масиву A[p,m,n]. Елементи кожного стовпчика окремо у кожному перерізі впорядковані за незменшенням.
12. Визначити знаходження та місцеположення заданого елемента X серед елементів кожної діагоналі окремо у всіх перерізах тривимірного масиву A[p,n,n]. Елементи кожного перерізу окремо впорядковані наскрізно по рядках за незбільшенням.
13. Визначити знаходження та місцеположення заданого елемента X серед елементів кожної діагоналі окремо у всіх перерізах тривимірного масиву A[p,n,n]. Елементи кожного

перерізу окремо впорядковані наскрізно по стовпчиках за незменшенням.

14. Визначити знаходження заданого елемента X серед елементів кожного перерізу тривимірного масиву A[p,m,n] окремо. Елементи кожного рядка окремо у кожному перерізі впорядковані за незбільшенням.
15. Визначити знаходження заданого елемента X серед елементів кожного перерізу тривимірного масиву A[p,m,n] окремо. Елементи кожного стовпчика окремо у кожному перерізі впорядковані за незменшенням.

Зауваження:

1) Першим елементом тривимірного масиву є A [1,1,1], останнім – A [p,m,n]. Зміна третього індексу відбувається швидше за інші, а першого – повільніше, ніж інших індексів.
2) У задачах, де треба брати перерізи тривимірного масиву A [p,m,n], їх треба брати по першому виміру (індексу). Тобто у масиві A [p,m,n] треба брати p перерізів.

МЕТОДИ (АЛГОРИТМИ) СОРТУВАННЯ

1. Алгоритм сортування №1 методу прямої вставки (з лінійним пошуком місця вставки від початку послідовності, що сортується , або «зліва»). Див. додаток 1, рис. №1.

2. Алгоритм сортування №2 методу прямої вставки (з лінійним пошуком місця вставки від елемента, що вставляється, або «справа», без бар'єру). Див. додаток 1, рис. №2.

3. Алгоритм сортування №3 методу прямої вставки (з лінійним пошуком місця вставки від елемента, що вставляється, або «справа», з бар'єром). Див. додаток 1, рис. №3.

4. Алгоритм сортування №4 методу прямої вставки (з двійковим пошуком місця вставки). Див. додаток 1, рис. №4.

5. Алгоритм сортування №1 методу прямого вибору. Див. додаток 1, рис. №5.

6. Алгоритм сортування №2 методу прямого вибору. Див. додаток 1, рис. №6.

7. Алгоритм сортування №3 методу прямого вибору. Див. додаток 1, рис. №7.

8. Алгоритм сортування №4 методу прямого вибору. Див. додаток 1, рис. №8.

9. Алгоритм сортування №5 методу прямого вибору. Див. додаток 1, рис. №9.

10. Алгоритм сортування №6 методу прямого вибору. Див. додаток 1, рис. №10.

11. Алгоритм сортування №7 методу прямого вибору. Див. додаток 1, рис. №11.

12. Алгоритм сортування №8 методу прямого вибору. Див. додаток 1, рис. №12.

13. Алгоритм сортування №1 методу прямого обміну (без модифікацій). Див. додаток 1, рис. №13.

14. Алгоритм сортування №2 методу прямого обміну (з використанням прапорця). Див. додаток 1, рис. №14.

15. Алгоритм сортування №3 методу прямого обміну (із запам'ятовуванням місця останньої перестановки). Див. додаток 1, рис. №15.

16. Алгоритм сортування №4 методу прямого обміну (Шейкерне сортування). Див. додаток 1, рис. №16.

17. Гібридний алгоритм "вставка – обмін". Див. додаток 1, рис. №17.

18. Гібридний алгоритм "вибір№1 – обмін". Див. додаток 1, рис. №18.

19. Гібридний алгоритм "вибір№2 – обмін". Див. додаток 1, рис. №19.

20. Гібридний алгоритм "вибір№3 – обмін". Див. додаток 1, рис. №20.

21. Гібридний алгоритм "вибір№4 – обмін". Див. додаток 1, рис. №21.

22. Сортування Шелла. Див. додаток 1, рис. №22. Кількість етапів та кроки між елементами на кожному етапі взяти в залежності від довжини послідовностей, що сортуються.

23.Швидке сортування. Див. додаток 1, рис. №23.

МЕТОДИ (АЛГОРИТМИ) ДВІЙКОВОГО ПОШУКУ

1. Двійковий пошук, що знаходить випадковий елемент з тих, що співпадають з шуканим елементом.
2. Двійковий пошук, що знаходить найлівіший елемент з тих, що співпадають з шуканим елементом.

СПОСОБИ ОБХОДУ (ВЗЯТТЯ ЕЛЕМЕНТІВ) МАСИВУ ДЛЯ ВИКОНАННЯ СОРТУВАННЯ

1. Переписати елементи заданого двовимірного масиву у додатковий одновимірний масив. Виконати сортування. Повернути результат у початковий масив.
2. Не використовуючи додаткового масиву, виконати сортування перетворюючи один індекс елементів "уявного" вектора у відповідні індекси елементів заданого двовимірного масиву.
3. Виконати сортування, здійснюючи обхід безпосередньо по елементах заданого двовимірного масиву, не використовуючи додаткових масивів і перетворень індексів.

4. Використовуючи елементи першого рядка кожного перерізу як ключі сортування, переставляти відповідні стовпчики кожен раз, коли треба переставляти ключі.
5. В якості першого етапу сортування сформувати додатковий вектор Sum, довжина якого дорівнює кількості стовпчиків і значеннями якого є суми елементів відповідних стовпчиків. Використовуючи елементи вектора Sum як ключі сортування, переставляти відповідні стовпчики кожен раз, коли треба переставляти ключі.
6. Використовуючи значення вектору перших елементів кожного перерізу A [*,1,1] як ключі сортування, переставляти відповідні перерізи кожен раз, коли треба переставляти ключі.
7. В якості першого етапу сортування сформувати додатковий вектор Sum, довжина якого дорівнює кількості перерізів і значеннями якого є суми елементів відповідних перерізів. Використовуючи елементи вектора Sum як ключі сортування, переставляти відповідні перерізи кожен раз, коли треба переставляти ключі.
8. У кожному перерізі спочатку повністю відсортувати елементи головної діагоналі, а потім побічної.

ВИПАДКИ ДЛЯ ЗАДАЧ СОРТУВННЯ

1. Елементи початкового масиву впорядковані відповідно до заданої ознаки.
2. Елементи початкового масиву невпорядковані.
3. Елементи початкового масиву впорядковані за протилежно заданою ознакою.

ВИПАДКИ ДЛЯ ЗАДАЧ ДВІЙКОВОГО ПОШУКУ

1. Потрібний елемент знаходиться після ¼ максимально можливого числа порівнянь в області пошуку.
2. Потрібний елемент знаходиться після ½ максимально можливого числа порівнянь в області пошуку.
3. Потрібний елемент знаходиться після ¾ максимально можливого числа порівнянь в області пошуку.
4. Потрібний елемент знаходиться при останньому порівнянні в області пошуку.
5. Потрібного елемента немає в області пошуку.

ВАРІАНТИ ІНДИВІДУАЛЬНИХ ЗАВДАНЬ

Початковий номер варіантів для крупи з шифром **КВ-XY** вираховується за формулою:

(25*X + 32*(Y – 1) + 1) mod 128

Варіант	Задача	Алгоритми	Способи обходу
1	3	1, 12, 22, 23	4
2	1	1	1, 2, 3
3	4	1, 9, 15, 22	5
4	2	21	1, 2, 3
5	5	1, 11, 16, 23	6
6	1	1, 2, 17	3
7	6	1, 12, 22, 23	7
8	2	5(обхід 3), 22(обхід 2), 23 (обхід 2)	2 або 3
9	7	4, 10, 14, 23	8
10	1	2	1, 2, 3
11	3	2, 11, 22, 23	4
12	2	20	1, 2, 3
13	4	2, 10, 16, 22	5
14	1	5, 6, 7, 8	3
15	5	2, 12, 15, 23	6
16	8	1, 2	–
17	6	2, 11, 22, 23	7
18	2	16, 22, 23	3
19	7	3, 9, 13, 23	8
20	1	3	1, 2, 3
21	3	3, 10, 22, 23	4
22	2	19	1, 2, 3
23	4	3, 11, 14, 22	5
24	1	5, 6, 18, 19	3
25	5	3, 9, 13, 23	6

26	2	2, 7, 14	3
27	6	3, 10, 22, 23	7
28	1	4	1, 2, 3
29	7	2, 12, 15, 23	8
30	2	18	1, 2, 3
31	3	4, 9, 22, 23	4
32	9	1, 2	—
33	4	4, 12, 13, 22	5
34	1	7, 8, 20, 21	3
35	5	4, 10, 14, 23	6
36	2	2, 6, 15	3
37	6	4, 9, 22, 23	7
38	1	5	1, 2, 3
39	7	1, 11, 16, 23	8
40	2	17	1, 2, 3
41	3	1, 11, 16, 23	4
42	1	13, 14, 15	3
43	4	1, 11, 16, 23	5
44	2	1, 8, 15	3
45	5	1, 12, 22, 23	6
46	6	1, 9, 15, 22	7
47	1	6	1, 2, 3
48	10	1, 2	—
49	7	4, 12, 13, 22	8
50	2	16	1, 2, 3
51	4	2, 12, 15, 23	5
52	1	13, 17, 18	3
53	3	4, 10, 14, 23	4
54	2	1, 5, 14	3
55	5	2, 11, 22, 23	6
56	6	2, 10, 16, 22	7
57	1	7	1, 2, 3
58	7	3, 11, 14, 22	8
59	2	15	1, 2, 3
60	5	3, 10, 22, 23	6
61	3	3, 9, 13, 23	4

62	2	13, 20, 21	3
63	4	3, 9, 13, 23	5
64	11	1, 2	—
65	1	13, 20, 21	3
66	1	8	1, 2, 3
67	6	3, 11, 14, 22	7
68	2	11	1, 2, 3
69	7	2, 10, 16, 22	8
70	1	1, 5, 14	3
71	3	2, 12, 15, 23	4
72	2	13, 17, 18	3
73	4	4, 10, 14, 23	5
74	1	12	1, 2, 3
75	5	4, 9, 22, 23	6
76	2	13	1, 2, 3
77	6	4, 12, 13, 22	7
78	1	1, 8, 15	3
79	7	1, 9, 15, 22	8
80	12	1, 2	—
81	3	1, 9, 15, 22	4
82	2	13, 14, 15	3
83	4	1, 12, 22, 23	5
84	1	14	1, 2, 3
85	5	1, 9, 15, 22	6
86	2	8	1, 2, 3
87	6	1, 11, 16, 23	7
88	1	2, 6, 15	3
89	7	4, 9, 22, 23	8
90	2	7, 8, 20, 21	3
91	3	2, 10, 16, 22	4
92	1	15	1, 2, 3
93	4	2, 11, 22, 23	5
94	2	7	1, 2, 3
95	5	2, 10, 16, 22	6
96	13	1, 2	—
97	6	2, 12, 15, 23	7

98	1	2, 7, 14	3
99	7	3, 10, 22, 23	8
100	2	5, 6, 18, 19	3
101	3	3, 11, 14, 22	4
102	1	16	1, 2, 3
103	2	6	1, 2, 3
104	1	16, 22, 23	3
105	4	3, 10, 22, 23	5
106	1	17	1, 2, 3
107	2	5	1, 2, 3
108	2	5, 6, 7, 8	3
109	5	3, 11, 14, 22	6
110	1	18	1, 2, 3
111	2	4	1, 2, 3
112	14	1, 2	–
113	6	3, 9, 13, 23	7
114	1	19	1, 2, 3
115	7	2, 11, 22, 23	8
116	1	5(обхід 3), 22(обхід 2), 23 (обхід 2)	2 або 3
117	3	4, 12, 13, 22	4
118	1	20	1, 2, 3
119	2	3	1, 2, 3
120	2	1, 2, 17	3
121	4	4, 9, 22, 23	5
122	1	21	1, 2, 3
123	5	4, 12, 13, 22	6
124	2	1	1, 2, 3
125	6	4, 10, 14, 23	7
126	2	2	1, 2, 3
127	7	1, 12, 22, 23	8
128	15	1, 2	–

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Основна література

1. Вирт Н. Алгоритмы и структуры данных. М.: Мир, 1989.
2. Марченко А.И., Марченко Л.А. Программирование в среде Turbo Pascal 7.0. – 9-е изд. – К.: Век+, Спб.: КОРОНА-Век, 2007. – 464 с., ил.
3. Т.Кормен, Ч.Лейзерзон, Р.Ривест. Алгоритмы: построение и анализ. М.: МЦНМО, 2000.
4. Кнут Д. Искусство программирования для ЭВМ. т.1. Основные алгоритмы., М.: Мир, 1976.
5. Кнут Д. Искусство программирования для ЭВМ. т.2. Получисленные алгоритмы., М.: Мир, 1977.
6. Кнут Д. Искусство программирования для ЭВМ. т.3. Сортировка и поиск., М.: Мир, 1978.

Додаткова література

7. Вирт Н. Алгоритмы + структуры данных = программы. М.:Мир, 1985.
8. Г.Буч. Объектно-ориентированное проектирование с примерами применения: Пер. с англ. – М.: Конкорд, 1992. – 519 с., ил.
9. Методичні вказівки для виконання лабораторних робіт з дисципліни „Програмування” для студентів спеціальностей „Комп’ютерні системи та мережі”, „Спеціалізовані комп’ютерні системи”, „Системне програмування” , ч. 1,2 (Укл. Р.Ф. Колінько, О.І. Марченко). Київ НТУУ „КПІ” – 2004, видавництво „Ювета”.

10. Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн, М.И. Задачи по программированию., М.: наука, 1988.

11. Martin J., McClure C. Structured techniques : The basis for CASE., 1988.

12. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов., М.: Мир, 1979.

13. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов., М.: Мир, 1981.

14. Дал У., Дейкстра., Хоор К. Структурное программирование., М.: Мир, 1975.

15. Дейкстра Э. Дисциплина программирования. М.: Мир, 1978.

16. Вирт Н. Систематическое программирование. Введение., М.: Мир,1977.

17. Лингер Р., Миллс Х., Уитт Б. Теория и практика структурного программирования., М.: Мир, 1982.

18. Брукс Ф.П. Как проектируются и создаются программные комплексы., М.: Наука, 1979.

ДОДАТОК 1.
АЛГОРИТМИ ДЛЯ ВИКОНАННЯ КУРСОВОЇ РОБОТИ

```
program Insert1;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   i, j, k : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   for i:=2 to n do
   begin
      B:=A[i];
      j:=1;
      while B>A[j] do j:=j+1;
      for k:=i-1 downto j do
            A[k+1]:=A[k];
      A[j]:=B;
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 1. Алгоритм сортування №1 методу прямої **вставки** (з лінійним пошуком місця вставки від початку послідовності, що сортується , або «зліва»)

```
program Insert2;
uses Crt;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   i, j : word;
Begin
   ClrScr;
   for i:=1 to n do read(A[i]);
   readln;

   for i:=2 to n do
   begin
     B:=A[i];
     j:=i;
      while (j>1) and (B<A[j-1]) do
      begin
            A[j]:=A[j-1];
            j:=j-1;
       end;
       A[j]:=B;
    end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 2. Алгоритм сортування №2 методу прямої **вставки** (з лінійним пошуком місця вставки від елемента, що вставляється, або «справа», без бар'єру).

```
program Insert3;
const n=10;
type
   TVector=array[0..n] of integer;
var
   A : TVector;
   i, j : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   for i:=2 to n do
   begin
     A[0]:=A[i];
     j:=i;
      while (j>1) and (A[0]<A[j-1]) do
      begin
            A[j]:=A[j-1];
            j:=j-1;
       end;
       A[j]:=A[0];
    end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 3. Алгоритм сортування №3 методу прямої **вставки** (з лінійним пошуком місця вставки від елемента, що вставляється, або «справа», з бар'єром).

```
program Insert4;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   X : integer;
   i, j, k, L, R : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   for i:=2 to n do
   begin
      X:=A[i];
      L:=1; R:=i;
      while L<R do
      begin
         j:=(L+R) div 2;
         if A[j] <= X then
            L:=j+1
         else
            R:=j;
      end;
      for k:=i-1 downto R do
            A[k+1]:=A[k];
      A[R]:=X;
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 4. Алгоритм сортування №4 методу прямої **вставки** (з двійковим пошуком місця вставки).

```
program Select1;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   Min : integer;
   s, i, imin : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   for s:=1 to n-1 do
   begin
      Min:=A[s]; imin:=s;
      for i:=s+1 to n do
         if A[i] < Min then
         begin
            Min:=A[i];
            imin:=i;
         end;
      A[imin]:=A[s];
      A[s]:=Min;
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 5. Алгоритм сортування №1 методу прямого **вибору**.

```
program Select2;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   s, i, imin : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   for s:=1 to n-1 do
   begin
      imin:=s;
      for i:=s+1 to n do
         if A[i] < A[imin] then
            imin:=i;
      B:=A[imin];
      A[imin]:=A[s];
      A[s]:=B;
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 6. Алгоритм сортування №2 методу прямого **вибору**.

```
program Select3;
uses Crt;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   Min, Max : integer;
   L, R, i, imin, imax : word;
Begin
   ClrScr;
   for i:=1 to n do read(A[i]);
   readln;

   L:=1; R:=n;
   while L<R do
   begin
      Min:=A[L]; imin:=L;
      Max:=A[L]; imax:=L;
      for i:=L+1 to R do
         if A[i] < Min then
         begin
            Min:=A[i];
            imin:=i;
         end
         else
            if A[i] > Max then
            begin
               Max:=A[i];
               imax:=i;
            end;

      A[imin]:=A[L];
      A[L]:=Min;
      if imax=L then A[imin]:=A[R]
                else A[imax]:=A[R];
      A[R]:=Max;

      L:=L+1; R:=R-1;
```

```
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 7. Алгоритм сортування №3 методу прямого **вибору**.

```
program Select4;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   L, R, i, imin, imax : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   L:=1; R:=n;
   while L<R do
   begin
      imin:=L; imax:=L;
      for i:=L+1 to R do
         if A[i] < A[imin] then
            imin:=i
         else
            if A[i] > A[imax] then
               imax:=i;

      B:=A[imin];
      A[imin]:=A[L];
      A[L]:=B;
      if imax=L then
      begin
         B:=A[imin];
```

```
            A[imin]:=A[R];
            A[R]:=B;
         end
         else begin
            B:=A[imax];
            A[imax]:=A[R];
            A[R]:=B;
         end;
         L:=L+1; R:=R-1;
      end;

      for i:=1 to n do write(A[i]:8);
      writeln;

End.
```

Рис. 8. Алгоритм сортування №4 методу прямого **вибору**.

```
program Select5;
uses Crt;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   Min : integer;
   s, i, imin : word;
Begin
   ClrScr;
   for i:=1 to n do read(A[i]);
   readln;

   for s:=1 to n-1 do
   begin
      Min:=A[s]; imin:=s;
      for i:=s+1 to n do
         if A[i] < Min then
         begin
            Min:=A[i];
            imin:=i;
         end;
      if imin<>s then
      begin
         A[imin]:=A[s];
         A[s]:=Min;
      end;
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 9. Алгоритм сортування №5 методу прямого **вибору**.

```
program Select6;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   s, i, imin : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   for s:=1 to n-1 do
   begin
      imin:=s;
      for i:=s+1 to n do
         if A[i] < A[imin] then
            imin:=i;

      if imin<>s then
      begin
         B:=A[imin];
         A[imin]:=A[s];
         A[s]:=B;
      end;
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 10. Алгоритм сортування №6 методу прямого **вибору**.

```
program Select7;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   Min, Max : integer;
   L, R, i, imin, imax : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   L:=1; R:=n;
   while L<R do
   begin
      Min:=A[L]; imin:=L;
      Max:=A[L]; imax:=L;
      for i:=L+1 to R do
         if A[i] < Min then
         begin
            Min:=A[i];
            imin:=i;
         end
         else
            if A[i] > Max then
            begin
               Max:=A[i];
               imax:=i;
            end;

      if imin<>L then
      begin
         A[imin]:=A[L];
         A[L]:=Min;
      end;
      if imax<>R then
      begin
         if imax=L then A[imin]:=A[R]
                   else A[imax]:=A[R];
```

```
         A[R]:=Max;
      end;
      L:=L+1; R:=R-1;
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 11. Алгоритм сортування №7 методу прямого **вибору**.

```
program Select8;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   L, R, i, imin, imax : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   L:=1; R:=n;
   while L<R do
   begin
      imin:=L; imax:=L;
      for i:=L+1 to R do
         if A[i] < A[imin] then
            imin:=i
         else
            if A[i] > A[imax] then
               imax:=i;

      if imin<>L then
      begin
         B:=A[imin];
         A[imin]:=A[L];
```

```
            A[L]:=B;
         end;
         if imax<>R then
            if imax=L then
            begin
               B:=A[imin];
               A[imin]:=A[R];
               A[R]:=B;
            end
            else begin
               B:=A[imax];
               A[imax]:=A[R];
               A[R]:=B;
            end;

         L:=L+1; R:=R-1;
      end;

      for i:=1 to n do write(A[i]:8);
      writeln;

End.
```

Рис. 12. Алгоритм сортування №8 методу прямого **вибору**.

```
program Exchange1;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   i, R : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   for R:=n downto 2 do
      for i:=1 to R-1 do
         if A[i] > A[i+1] then
         begin
            B:=A[i];
            A[i]:=A[i+1];
            A[i+1]:=B;
         end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 13. Алгоритм сортування №1 методу прямого **обміну** (без модифікацій).

```
program Exchange2;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   i, R : word;
   f : boolean;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   R:=n; f:=True;
   while f=True do
   begin
      f:=False;
      for i:=1 to R-1 do
         if A[i] > A[i+1] then
         begin
            B:=A[i];
            A[i]:=A[i+1];
            A[i+1]:=B;
            f:=True;
         end;
      R:=R-1;
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 14. Алгоритм сортування №2 методу прямого **обміну** (з використанням прапорця).

```
program Exchange3;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   i, R, k : word;

Begin
   for i:=1 to n do read(A[i]);
   readln;

   R:=n;
   while R>1 do
   begin
      k:=1;
      for i:=1 to R-1 do
         if A[i] > A[i+1] then
         begin
            B:=A[i];
            A[i]:=A[i+1];
            A[i+1]:=B;
            k:=i;
         end;
      R:=k;
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 15. Алгоритм сортування №3 методу прямого **обміну** (із запам'ятовуванням місця останньої перестановки).

```
program Exchange4;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   i, L, R, k : word;
Begin
   for i:=1 to n do read(A[i]); readln;

   L:=1; R:=n; k:=1;
   while L<R do
   begin
      for i:=L to R-1 do
         if A[i] > A[i+1] then
         begin
            B:=A[i];
            A[i]:=A[i+1];
            A[i+1]:=B;
            k:=i;
         end;
      R:=k;
      for i:=R-1 downto L do
         if A[i] > A[i+1] then
         begin
            B:=A[i];
            A[i]:=A[i+1];
            A[i+1]:=B;
            k:=i;
         end;
      L:=k+1;
   end;
   for i:=1 to n do write(A[i]:8); writeln;
End.
```

Рис. 16. Алгоритм сортування №4 методу прямого **обміну** (Шейкерне сортування).

```
program InsertExchange;
```

```
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   i, j : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   for i:=2 to n do
   begin
      j:=i;
      while (j>1) and (A[j]<A[j-1]) do
      begin
            B:=A[j];
            A[j]:=A[j-1];
            A[j-1]:=B;
            j:=j-1;
       end;
    end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 17. Гібридний алгоритм "вставка – обмін".

```
program SelectExchange1;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   Min : integer;
   s, i : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   for s:=1 to n-1 do
   begin
      Min:=A[s];
      for i:=s+1 to n do
         if A[i] < Min then
         begin
            Min:=A[i];
            A[i]:=A[s];
            A[s]:=Min;
         end;
    end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 18. Гібридний алгоритм "вибір№1 – обмін".

```
program SelectExchange2;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   B : integer;
   s, i : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;

   for s:=1 to n-1 do
   begin
      for i:=s+1 to n do
         if A[i] < A[s] then
         begin
            B:=A[i];
            A[i]:=A[s];
            A[s]:=B;
         end;
   end;

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 19. Гібридний алгоритм "вибір№2 – обмін".

```
program SelectExchange3;
const n=10;
type  TVector=array[1..n] of integer;
var   A : TVector;
      Min, Max, B : integer;
      L, R, i : word;
Begin
   for i:=1 to n do read(A[i]); readln;
   L:=1; R:=n;
   while L<R do
   begin
      if A[L] > A[R] then
      begin
         B:=A[L];
         A[L]:=A[R];
         A[R]:=B;
      end;
      Min:=A[L]; Max:=A[R];
      for i:=L+1 to R-1 do
         if A[i] < Min then
         begin
            Min:=A[i];
            A[i]:=A[L];
            A[L]:=Min;
         end
         else
            if A[i] > Max then
            begin
               Max:=A[i];
               A[i]:=A[R];
               A[R]:=Max;
            end;
      L:=L+1; R:=R-1;
   end;
   for i:=1 to n do write(A[i]:8); writeln;
End.
```

Рис. 20. Гібридний алгоритм "вибір№3 – обмін".

```
program SelectExchange4;
uses Crt;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   Min, Max : integer;
   L, R, i : word;
Begin
   ClrScr;
   for i:=1 to n do read(A[i]); readln;
   L:=1; R:=n;
   while L<R do
   begin
      for i:=L to R do
         if A[i] < A[L] then
         begin
            Min:=A[i];
            A[i]:=A[L];
            A[L]:=Min;
         end
         else
            if A[i] > A[R] then
            begin
               Max:=A[i];
               A[i]:=A[R];
               A[R]:=Max;
            end;

      L:=L+1; R:=R-1;
   end;
   for i:=1 to n do write(A[i]:8); writeln;

End.
```

Рис. 21. Гібридний алгоритм "вибір№4 – обмін".

```
program Shell;
uses Crt;
const n = 10;  max_t = (n-1) div 4 + 1;
type
   TVector=array[1..n] of integer;
   TStages=array[1..max_t] of integer;
var
   A : TVector;
   H : TStages;
   B : integer;
   i, j, k, p, t : word;
Begin
   ClrScr;
   for i:=1 to n do read(A[i]); readln;

   if n<4 then t := 1
          else t := Trunc(Ln(n)/Ln(2))-1;
   H[t]:=1;
   for i:=t-1 downto 1 do H[i]:=2*H[i+1]+1;
   for p:= 1 to t do
   begin
      k:=H[p];
      for i:=k+1 to n do
      begin
         B:=A[i];
         j:=i;
         while (j>k) and (B<A[j-k]) do
         begin
            A[j]:=A[j-k];
            j:=j-k;
         end;
         A[j]:=B;
      end;
   end;
   for i:=1 to n do write(A[i]:8); writeln;
End.
```

Рис. 22. Сортування Шелла.

```
program QuickSort;
uses Crt;
const n = 10;
type  TVector=array[1..n] of integer;
var   A : TVector;
      i : word;
procedure QSort (L, R : word);
var
   B, Tmp : integer;
   i, j : word;
begin
   B:=A[(L+R) div 2]; i:=L; j:=R;
   while i<=j do
   begin
      while A[i] < B do i:=i+1;
      while A[j] > B do j:=j-1;
      if i<=j then
      begin
        Tmp:=A[i];
        A[i]:=A[j];
        A[j]:=Tmp;
        i:=i+1;
        j:=j-1;
      end;
   end;
   if L<j then QSort(L,j);
   if i<R then QSort(i,R);
end;
Begin
   ClrScr;
   for i:=1 to n do read(A[i]); readln;

   QSort(1,n);

   for i:=1 to n do write(A[i]:8); writeln;
End.
```

Рис. 23. Швидке сортування.

```
program BinSearch1;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   X : integer;
   i, L, R : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;
   readln(X);

   L:=1; R:=n;
   while L<=R do
   begin
      i:=(L+R) div 2;
      if A[i] = X then Break
      else
         if A[i] < X then
            L:=i+1
         else
            R:=i-1;
   end;
   if L<=R then writeln('Element ', X:2, '  has been
found at ', i:2, '  position')
   else writeln('Element has not been found');

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 24. Двійковий пошук, що знаходить випадковий елемент з тих, що співпадають з шуканим елементом (Алгоритм №1).

```
program BinSearch2;
const n=10;
type
   TVector=array[1..n] of integer;
var
   A : TVector;
   X : integer;
   i, L, R : word;
Begin
   for i:=1 to n do read(A[i]);
   readln;
   readln(X);

   L:=1; R:=n;
   while L<R do
   begin
      i:=(L+R) div 2;
      if A[i] < X then
         L:=i+1
      else
         R:=i;
   end;
   if A[R]=X then writeln('Element ', X:2, '  has
been found at ', R:2, '  position')
   else writeln('Element has not been found');

   for i:=1 to n do write(A[i]:8);
   writeln;

End.
```

Рис. 25. Двійковий пошук, що знаходить найлівіший елемент з тих, що співпадають з шуканим елементом (Алгоритм №2).

ДОДАТОК 2. ПЕРШИЙ ТИТУЛЬНИЙ ЛИСТ КУРСОВОЇ РОБОТИ.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

КУРСОВА РОБОТА

з дисципліни "Структури даних і алгоритми"

Виконав: Прізвище Ініціали

Група: КВ-??

Номер залікової книжки: КВ-????

Допущений до захисту

_____

2 семестр 20__/20__ навч.року

ДОДАТОК 3. ДРУГИЙ ТИТУЛЬНИЙ ЛИСТ КУРСОВОЇ РОБОТИ.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Узгоджено

Керівник роботи

_____/Марченко О.І./

ЗАХИЩЕНА "__"_____20__р.

з оцінкою_____

_____ /Марченко О.І./

Дослідження ефективності методів сортування (назви конкретних методів сортування)на багатовимірних масивах

Виконавець роботи:_____

(підпис)

Прізвище Ім'я По батькові

_____20__ р.

ДОДАТОК 4. СТРУКТУРА ТЕХНІЧНОГО ЗАВДАННЯ КУРСОВОЇ РОБОТИ.

ТЕХНІЧНЕ ЗАВДАННЯ НА КУРСОВУ РОБОТУ

I. Описати принцип та схему роботи кожного із досліджуваних методів сортування або пошуку для одновимірного масиву.

II. Скласти алгоритми сортування або пошуку в багатовимірному масиві заданими методами, згідно до варіанту, та написати відповідну програму на мові програмування.

Програма повинна задовольняти наступним вимогам:

1) Всі алгоритми повинні бути реалізовані в рамках ОДНІЄЇ програми з діалоговим інтерфейсом для вибору варіантів тестування та виміру часу кожного алгоритму.
2) Одним з варіантів запуску програми має бути режим запуску виміру часу всіх алгоритмів у пакетному режимі, тобто запуск всіх алгоритмів для всіх випадків і побудова результуючої таблиці за наведеним нижче зразком для масиву з заданими геометричними розмірами.
3) При реалізації програми повинні бути використані модулі (unit).
4) Програма повинна мати коментарі для всіх структур даних, процедур та функцій, а також до основних смислових фрагментів алгоритмів.

III. Виконати налагодження та тестування коректності роботи написаної програми.

IV. Провести практичні дослідження швидкодії складених алгоритмів.

V. За результатами досліджень скласти порівняльні таблиці за різними ознаками.

Для виконання ґрунтовного аналізу алгоритмів потрібно виконати виміри часу та побудувати таблиці для декількох масивів з різними геометричними розмірами, а також для стандартного випадку одномірного масиву. Кількість необхідних таблиць для масивів з різними геометричними розмірами залежить від задачі конкретного варіанту курсової роботи і вибирається самостійно студентом так, щоб виконати всебічний та ґрунтовний порівняльний аналіз заданих алгоритмів. За отриманими результатами можуть бути також побудовані графіки для наочності подання інформації.

VI. Виконати порівняльний аналіз поведінки заданих алгоритмів за отриманими результатами:

- для одномірного масиву відносно загальновідомої теорії;
- для багатовимірних масивів відносно результатів для одномірного масиву;
- для заданих алгоритмів на багатовимірних масивах між собою;
- дослідити вплив різних геометричних розмірів багатовимірних масивів на поведінку алгоритмів та їх взаємовідношення між собою;

- для всіх вищезазначених пунктів порівняльного аналізу пояснити, ЧОМУ алгоритми в розглянутих ситуаціях поводять себе саме так, а не інакше.

VII. Зробити висновки за виконаним порівняльним аналізом.

VIII. Програму курсової роботи під час її захисту ОБОВЯЗКОВО мати при собі на електронному носії інформації.

Варіант № ___

Задача

Умова задачі за варіантом.

Досліджувані методи та алгоритми

Перелік методів та алгоритмів за варіантом.

Способи обходу

Перелік способів обходу за варіантом (якщо є).

Випадки дослідження

Перелік випадки дослідження, які потрібно розглянути за варіантом.