МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних та лабораторних робот

«Основи програмування мовою С++. Використання алгоритмів сортування»

з курсів
«Інформатика», «Основи інформаційних технологій»,
«Структури і алгоритми обробки даних»
для студентів спеціальностей: 152 «Метрологія та вимірювальна техніка»,
172 «Телекомунікація та радіотехніка» та 151 «Автоматизація та комп'ютерно-інтегровані технології»
денної та заочної форм навчання

Затверджено
редакційно-видавничою
радою університету,
протокол №1 від 19.02.2020

Харків
НТУ «ХПІ»
2020

Методичні вказівки до виконання практичних і лабораторних робіт «Основи програмування мовою С++. Використання алгоритмів сортування» з курсу «Інформатика», «Основи інформаційних технологій», «Структури і алгоритми обробки даних» для студентів спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології», 152 «Метрологія та вимірювальна техніка», 172 «Телекомунікація та радіотехніка» денної та заочної форм навчання / уклад. О. Є. Тверитникова, В.А. Крилова. – Харків : НТУ «ХПІ». – 26 с.

Укладачі О. Є. Тверитникова
В.А. Крилова

Рецензент д.т.н., професор Б.М. Горкунов

Кафедра інформаційно-вимірювальні технології і системи

ЗМІСТ

ВСТУП

Розвиток сучасних технологій неможливо без використання комп'ютерної техніки та програмного забезпечення. Підготовка фахівців в області автоматики, вимірювальної та медичної техніки вимагає великих знань і навичок володіння обчислювальною технікою, а так само знання основ алгоритмізації та програмування на мовах високого рівня таких як C/C ++.

Методичні вказівки призначені для вивчення мови програмування C++ на практичних заняттях та лабораторних роботах, а також для самостійного освоєння. У методичних вказівках детально і доступно розглянуті синтаксис, семантика і техніка програмування на мові C ++ з використанням алгоритмів сортування даних. Описано всі етапи проектування програм, наведені докладні коментарі програмного коду, проаналізовані результати обчислень, показані типові проблеми та шляхи їх вирішення. Велика увага приділяється алгоритмам ініціалізації, пошуку, сортування та прикладів розв'язання задач з використанням різних методів та алгоритмів сортування масивів даних. Наведено індивідуальні завдання для виконання практичних та лабораторних робіт, також приведені і завдання для самостійного вивчення основ алгоритмізації та програмування на мові C++. Розглянуті приклади і завдання, допоможуть ефективному освоєнню основ програмування для учнів і початківців програмістів.

ЛАБОРАТОРНА РОБОТА

ПРОГРАМУВАННЯ З ВИКОРИСТАННЯМ АЛГОРИТМІВ СОРТУВАННЯ

Мета роботи – отримання знань і навичок, необхідних для виконання сортування даних, ознайомлення з базовими методами та алгоритмами сортування, використання їх на практиці в процесі розроблення програм мовою програмування С++.

Загальні теоретичні відомості

При роботі з масивами найчастіше доводиться виконувати операції сортування даних. Від ефективності реалізації цих операцій часто залежить ефективність всієї програми.

Сортування – впорядковування за ключовими елементами структури даних, на якій визначено відношення порядку.

Алгоритмом сортування називається алгоритм для впорядкування деяких елементів множини. Зазвичай алгоритмом сортування вважають алгоритм упорядкування елементів за зростанням або спаданням.

Практично кожен алгоритм сортування можна розбити на три основних етапи:

I. Порівняння (впорядкованість пари елементів).

II. Перестановка (зміна місць пари елементів).

III. Перестановка (власне алгоритм сортування).

Алгоритми сортування мають велике практичне застосування в процесі оброблення, збереження великих обсягів інформації. Незважаючи на значну кількість алгоритмів сортування – універсального алгоритму, здатного для вирішення більшої кількості завдань не існує. Вибір алгоритму сортування для конкретного завдання є досить складний процес, який залежить від різних параметрів.

До основних параметрів алгоритмів сортування належить: час сортування, необхідна пам'ять, стійкість, природність поведінки

Час сортування – основний параметр, що характеризує швидкодію алгоритму.

Необхідна пам'ять – один з параметрів, який характеризується тим, що ряд алгоритмів сортування вимагають виділення додаткової пам'яті для тимчасового зберігання даних.

При оцінці пам'яті що використовується не буде враховуватися місце, яке займає вихідний масив даних та витрати які не залежать від вхідної послідовності, наприклад, на зберігання коду програми.

Стійкість – це параметр, який відповідає за те, що сортування не змінює взаємного розташування рівних елементів.

Природність поведінки – параметр, якій вказує на ефективність методу при обробленні вже відсортованих, або частково відсортованих даних. Алгоритм поводиться природно, як що враховує цю характеристику вхідної послідовності і працює краще.

Крім загальнонаукового інтересу до алгоритмів сортування, в кожному алгоритмі потрібно оцінити його складність. Під складністю алгоритму розуміється максимальне число елементарних кроків алгоритму. На прикладах сортувань можна показати, як шляхом ускладнення алгоритму, хоча під рукою і є вже очевидні методи, можна домогтися значного виграшу в ефективності.

При вирішенні задачі сортування масивів зазвичай висувається вимога мінімального використання додаткової пам'яті, з якого витікає неприпустимість використання додаткових масивів.

Для оцінки швидкодії алгоритмів різних методів сортування, як правило, використовують два показники: кількість присвоювань та кількість порівнянь.

Існує багато алгоритмів сортування, серед яких можна розглянути *сортування вибором, обміном, вставками, Шелла, злиттям, підрахунком* тощо. Їх усі можна розбити на дві великі групи залежно від того, де містяться вихідні дані: в оперативній (внутрішній) пам’яті комп’ютера чи на зовнішніх носіях (у зовнішній пам’яті). Традиційно їх так і називають: *внутрішнє та зовнішнє сортування*.

Усі алгоритми сортування також поділяються на базові або прямі – ті, що працюють порівняно повільно, проте не потребують великих зусиль для реалізації й відлагодження – та швидкі або удосконалені методи – ті, що сортують великі масиви даних досить швидко, але зазвичай мають

достатньо складний для розуміння й громіздкий алгоритм. Удосконалені методи сортування базуються на тих самих принципах, що й прямі, але використовують деякі оригінальні ідеї для збільшення швидкості процесу впорядкування. Прямі методи на практиці використовуються не часто, оскільки мають відносно низьку швидкодію. Однак вони добре показують суть основаних на них удосконалених методів.

Прямі методи сортування за принципом, який лежить в основі методу, у свою чергу діляться на три підгрупи:

– сортування простими вставками (включенням);
– сортування вибором (виділенням);
– сортування обміном («бульбашкове»).

Метод обмінного сортування

Метод обмінного сортування (бульбашковий) ще називається «бульбашковим» методом, один з найпростіших методів сортування. Основні позитивні риси методу – це легка реалізація у вигляді програми.

Алгоритм складається в повторюваних проходах посортованого масиву. За кожен прохід елементи послідовно порівнюються попарно і, якщо порядок в парі невірний, виконується обмін елементів.

Проходи по масиву повторюються до тих пір, поки на черговому проході не опиниться, що обміни більше не потрібні, що означає – масив відсортований. При проході алгоритму, елемент, що стоїть не на своєму місці, «спливає» до потрібної позиції як бульбашка у воді, звідси і назва алгоритму. Алгоритм сортування методом бульбашки:

1. Порівнюємо перший і другий елементи масиву. Якщо перший елемент більший, ніж другий, то міняємо їх місцями.

2. Порівнюємо другий і третій елементи масиву. Якщо другий елемент більший, ніж третій, то міняємо їх місцями.

3. ...

4. Порівнюємо передостанній (N-1) і останній (N) елементи масиву. Якщо передостанній елемент більший, ніж останній, то міняємо їх місцями. Схема алгоритму представлена на рис. 1.

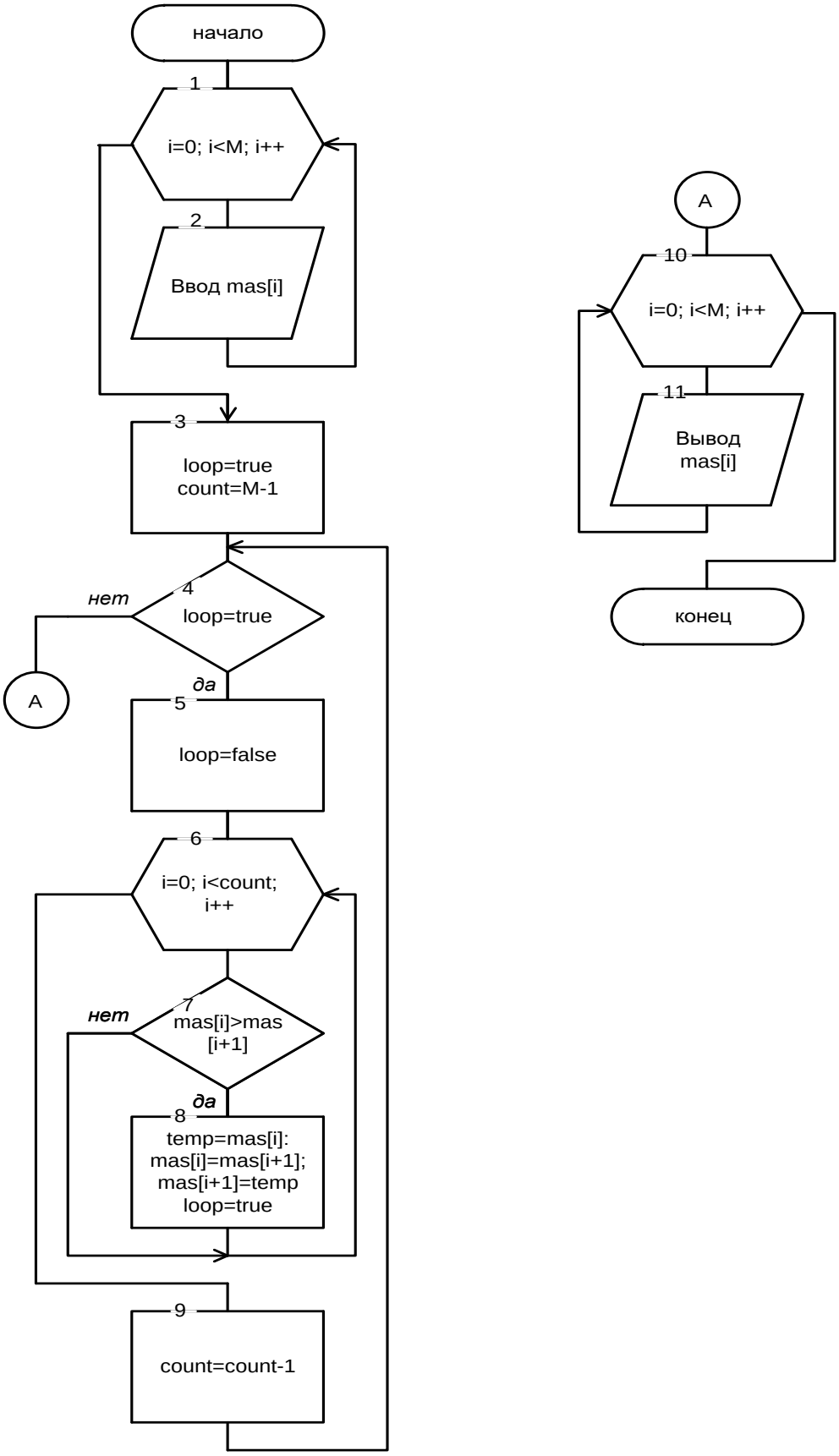

Рисунок 1 – Алгоритм програми сортування одновимірного масиву

Програма, що реалізує представлений алгоритм:

```
#include <iostream>
#define M 10
void main()
{
	float mas[M], temp;
	for(Int i = 0; i <M; ++ i)
		cin >> mas[i];
	bool loop = true;
	int count = M – 1;
	while (loop)
	{
		loop = false;
		for (int i = 0; i <count; ++ i)
			if (mas0[i]> mas [i + 1])
			{
				temp = mas[i];
				mas[i] = mas [i + 1];
				mas[i + 1] = temp;
				loop = true;
			}
		count–;
	}
	for (int i = 0; i <M; ++ i)
		cout << mas[i] << '\ n';
}
```

У програмі сортування проводиться в масиві *mas* [*M*]. В результаті роботи програми елементи масиву будуть впорядковані за зростанням їх значень. Слід також зазначити, що в результаті кожного перебору масиву максимальний елемент стає на своє місце. Тому для скорочення числа операцій використовується змінна *count*, яка має початкове значення, рівне розміру масиву, але після виконання кожного проходу зменшується на 1, зменшуючи тим самим довжину масиву. У найгіршому разі упорядковуватися буде масив, що складається з двох елементів. Може

скластися ситуація, коли масив довжиною більше двох елементів вже впорядкований. При проході за таким масиву не буде виконано жодного обміну, і процедуру сортування можна зупинити. Для визначення такої ситуації використовується логічна змінна *loop*.

Метод сортування вставками

Метод сортування вставками. Цей метод сортування менш ефективний, ніж більш складні алгоритми (наприклад, алгоритм швидкого сортування). Між тим, метод має низку переваг – простота реалізації, ефективність на невеликих наборах даних, ефективність на частково впорядкованих даних, стійкість (алгоритм не міняє порядок елементів, які вже відсортовані), можливість виконання сортування вхідної послідовності даних (наприклад, при отриманні даних в телекомунікації, в кожен момент часу входить послідовність є відсортованої, тоді як для обмінного сортування, якщо в послідовність додаються дані, то сортування повинна бути виконана ще раз повністю).

Суть методу полягає в тому, що на кожному кроці алгоритму вибирається один з елементів вхідних даних і вставляється його на потрібну позицію в уже відсортованому списку, до тих пір, поки набір вхідних даних не буде вичерпаний (рис. 2).

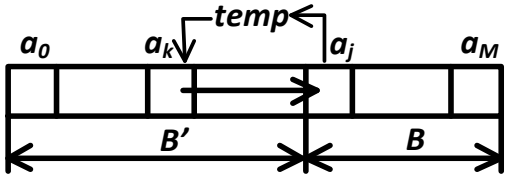

Рисунок 2 – Схема сортування методом простої вставки

Вважаємо, що на певному етапі виконання сортування масиву є послідовність B' з j–1 відсортованих значень. Елемент a_j масиву порівнюється послідовно з кожним із значень масиву від a_1 до a_{j-1} і поміщається на своє місце в відсортованої послідовності. На практиці це означає що, якщо елемент повинен бути розміщений в масиві в k–тій позиції, то він буде розміщений в цій позицію. При цьому впорядковані елементи масиву $a_k..a_{j-1}$ будуть зміщені в позиції $a_k + 1..a_j$. Далі

виконуються ті ж операції для елементів масиву $a_j + 1$, $a_j + 2$, ... поки набір вхідних даних не буде вичерпаний (рис. 2).

Вважаємо, що на певному етапі виконання сортування масиву є послідовність B' з j–1 відсортованих значень. Елемент a_j масиву порівнюється послідовно з кожним із значень масиву від a_1 до a_{j-1} і поміщається на своє місце в відсортованої послідовності.

На практиці це означає що, якщо елемент повинен бути розміщений в масиві в k–тій позиції, то він буде розміщений в цій позицію. При цьому впорядковані елементи масиву $a_k..a_{j-1}$ будуть зміщені в позиції $a_k + 1.. a_j$.

Далі виконуються ті ж операції для елементів масиву $a_j + 1$, $a_j + 2$, ... поки набір вхідних даних не буде вичерпаний (рис. 3).

Вважаємо, що на певному етапі виконання сортування масиву є послідовність B' з j–1 відсортованих значень. Елемент a_j масиву порівнюється послідовно з кожним із значень масиву від a_1 до a_{j-1} і поміщається на своє місце в відсортованої послідовності. На практиці це означає що, якщо елемент повинен бути розміщений в масиві в k–тій позиції, то він буде розміщений в цій позицію.

При цьому впорядковані елементи масиву $a_k..a_{j-1}$ будуть зміщені в позиції $a_k + 1.. a_j$. Далі виконуються ті ж операції для елементів масиву a_j $j + 1$, $a_j + 2$, ... Елемент a_j масиву порівнюється послідовно з кожним із значень масиву від a_1 до a_{j-1} і поміщається на своє місце в відсортованої послідовності. На практиці це означає що, якщо елемент повинен бути розміщений в масиві в k–тій позиції, то він буде розміщений в цій позицію. При цьому впорядковані елементи масиву $a_k..a_{j-1}$ будуть зміщені в позиції $a_k + 1.. a_j$.

Далі виконуються ті ж операції для елементів масиву $a_j + 1$, $a_j + 2$, ... Елемент aj масиву порівнюється послідовно з кожним із значень масиву від a_1 до a_{j-1} і поміщається на своє місце в відсортованої послідовності. На практиці це означає що, якщо елемент повинен бути розміщений в масиві в k–тій позиції, то він буде розміщений в цій позицію.

При цьому впорядковані елементи масиву $a_k..a_{j-1}$ будуть зміщені в позиції $a_k + 1.. a_j$. Далі виконуються ті ж операції для елементів масиву $a_j + 1$, $a_j + 2$, ...

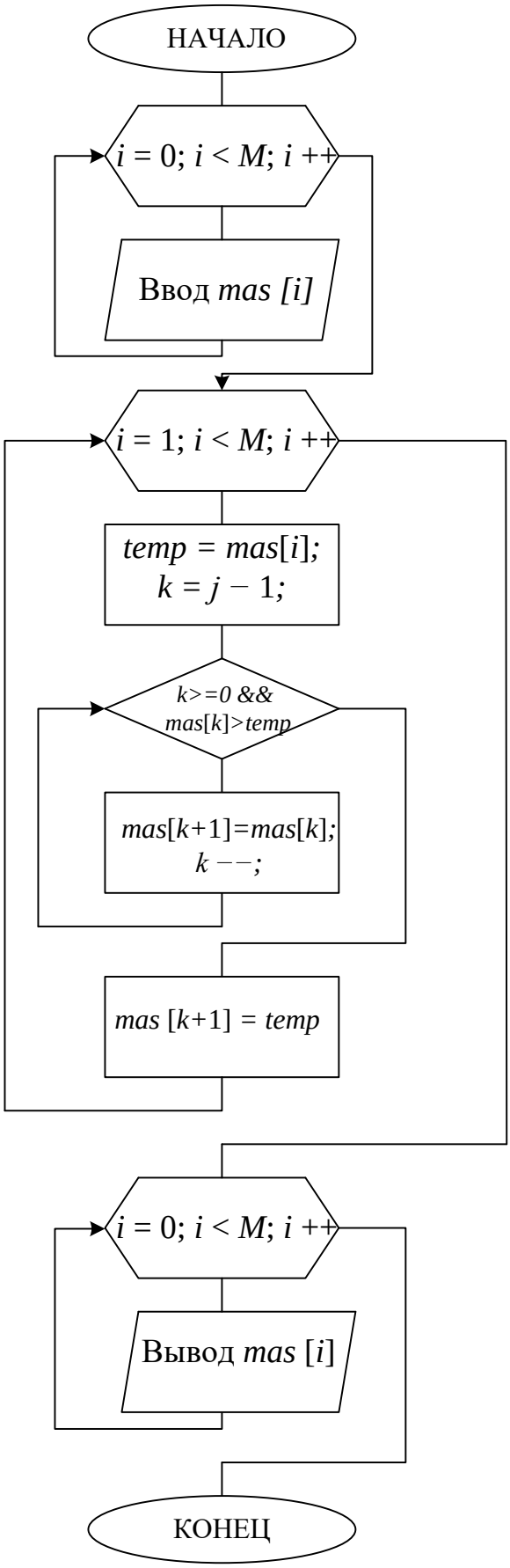

Рисунок 3 – Блок–схема алгоритму сортування методом простої вставки

Програма, що реалізує представлений алгоритм

```
#include <iostream>
#define M 5
void main ()
{
	float mas[M], temp;
	int k;
	for (int i = 0; i <M; ++ i)
		cin >> mas[i];
	for (int j = 1; j <M; ++ j)
	{
		temp = mas[j];
		k = j –1;
		while((k> = 0) && (mas [k]> temp))
		{
			mas[k + 1] = mas [k];
			k–;
		}
		mas[k + 1] = temp;
	}
	for (int i = 0; i <M; ++ i)
		cout << mas[i] << '\ n';
}
```

Метод сортування вибором

Метод сортування вибором. Ідея методу полягає в тому, щоб створювати відсортовану послідовність шляхом приєднання до неї у правильному порядку одного елемента за іншим. Побудова відсортованої послідовності починається з лівого кінця масиву. Алгоритм складається з n послідовних кроків, починаючи від нульового і закінчуючи (n-1)-м. На i -му кроці вибирається найменший з елементів $a[i]...a[n-1]$ і виконується заміна його місцями з $a[i]$. Незалежно від номера поточного кроку i, в результаті його виконання послідовність $a[0]...a[i\ ]$ є впорядкованою. Таким чином, на (n-1)-му кроці вся послідовність, окрім $a[n]$, виявляється

відсортованою, а $a[n]$ стоїть справедливо на останньому місці: всі менші елементи вже розташовані зліва.

Метод сортування вибором характеризується квадратичною залежністю часу сортування t від кількості елементів:

$$t = a * N2 + b * N * lgN$$

де a і b – константи, що залежать від програмної реалізації алгоритму.

Іншими словами, для сортування масив потрібно переглянути N раз.

Ідея алгоритму полягає в наступному. Він полягає в тому що з масиву вибирається найменший елемент і міняється місцями з першим елементом, потім розглядаються елементи, починаючи з другого, і найменший з них міняється місцями з другим елементом і так далі n–1 раз (при останньому проході циклу при необхідності міняються місцями передостанній і останній елементи масиву).

Наприклад є вихідна несортованими послідовність a [0..n–1]. Для сортування за зростанням масиву вибираємо з нього найменший елемент і ставимо його на перше місце. Тобто міняємо знайдений найменший елемент і перший. Потім в послідовності починаючи з 2–го елемента і до кінця – аналогічно шукаємо найменший елемент і міняємо його місцями з 2–м. І так далі до передостаннього елемента. Сортування закінчується, коли в залишилася невідсортоване послідовності залишається один елемент. Таким чином, алгоритм має n–1 ітерацій. І на кожній i (= [0..n–1]) вибирається найменший і міняється місцями з x[i]. Коли i=n–1 сортування припиняється, і ми отримаємо відсортовану послідовність, блок–схема алгоритму програми представлена на рис. 4.

Наприклад, є послідовність: 7, 2, 6, 5, 10, 4. n = 6.

0 крок i = 0, min = 2 (i = 1) міняємо місцями x[0] = 7 і x[1] = 2
2, 7, 6, 5, 10, 4.

1 крок i = 1, min = 4 (i = 5) міняємо місцями x[1] = 7 і x[5] = 4
2, 4, 6, 5, 10, 7.

2 крок i = 2, min = 5 (i = 3) міняємо місцями x[2] = 6 і x[3] = 5
2, 4, 5, 6, 10, 7.

3 крок i = 3, min = 6 (i = 3) міняємо місцями x[3] = 6 і x[3] = 6
2, 4, 5, 6, 10, 7.

4 крок i = 4, min = 4 (i = 5) міняємо місцями x[4] = 10 і x[5] = 7
2, 4, 5, 6, 7, 10.

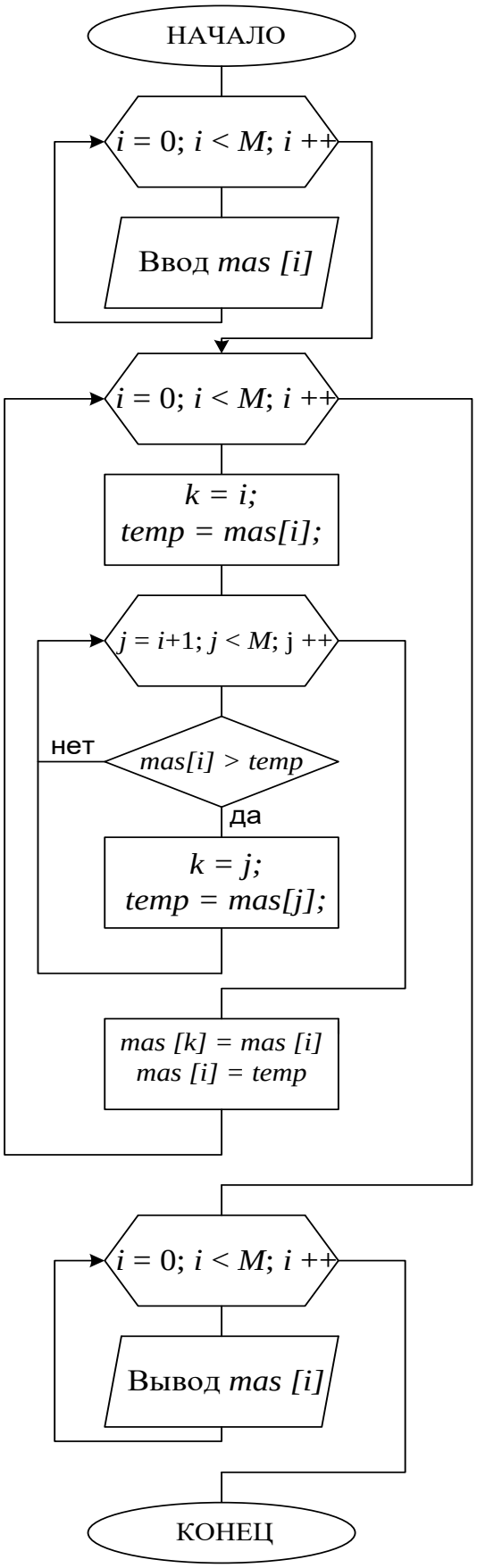

Рисунок 4 – Алгоритм сортування методом вибору

Програма, що реалізує представлений алгоритм

```
#include <iostream>
#define M 5
void main ()
{
        float mas[M], temp;
        int k, I, j;
        for (int i = 0; i <M; ++ i)
                cin >> mas[I];
        for (I = 0; i <M; i ++)
        {
                k = I;
                temp = mas [i];
                for (j= i + 1; j <M; j ++)
                {
                        if (mas [j] <temp)
                        {
                                k = j;
                                temp = mas [j];     }
                }
                mas [k] = mas [i];
                mas [i] = temp;
        }
        for (int i = 0; i <M; ++ i)
                cout << mas[i];
}
```

Метод сортування Шелла

Метод сортування Шелла. Сортування Шелла є модифікацією алгоритму сортування простими вставками.

Розглянемо алгоритм сортування на прикладі масиву *a*[0].. *a*[15]:

1. Виконати сортування простими вставками кожних 8 груп з 2-х елементів (*a*[0], *a*[8]), (*a*[1], *a*[9]) ... (*a*[7], *a*[15]).

2. Виконати сортування кожної з чотирьох груп по 4 елементи (a[0], *a*[4], *a*[8], *a*[12]) ... (*a*[3], *a*[7], *a*[11], *a*[15]).

3. Виконати сортування 2 груп по 8 елементів, починаючи з (*a*[0], *a*[2], *a*[4], *a*[6], *a*[8], *a*[10], *a*[12], *a*[14]).

4. Виконати сортування вставками всіх 16 елементів.

Метод сортування злиттям

Метод сортування злиттям. При сортуванні злиттям на *i*-му кроці масив *a* розбивається на впорядковані підмасиви (послідовності елементів масиву, що розташовані поспіль) довжини *Size*. Пари сусідніх підмасивів зливаються у впорядковані підмасиви довжини *Size**2 у допоміжному масиві *b*. Після присвоєння *a* значення *b* та збільшення вдвічі значення *Size* злиття повторюється для підмасивів більшого розміру. Процес завершується, коли *Size* стає більшим або рівним *n*. Оскільки при *i* = 1 підмасиви з 1 елемента впорядковані за означенням, у результаті буде отримано впорядкований масив a.

Метод сортування підрахунком

Метод сортування підрахунком. В сортуванні підрахунком передбачається, що всі *n* елементів – цілі числа, які лежать в інтервалі від 0 до *k*, де *k* – деяка ціла стала. Основна ідея сортування підрахунком полягає в тому, щоб для кожного вхідного елементу *x* визначити кількість елементів, які менші за *x*. За допомогою даної інформації елемент *x* можна розташувати на тій позиції вихідного масиву, де він повинен знаходитись. Наприклад, якщо всього є 17 елементів, які менші за *x*, то у вихідній послідовності елемент x повинен займати 18-у позицію.

Порядок виконання лабораторної роботи

1. Відповідно за номером по журналу виберіть індивідуальне завдання.

2. Розробіть алгоритм вирішення завдання.

3. Складіть текст програми.

4. Створіть проект в інтегрованому середовищі розробки Microsoft *Visual Studio* (*lab№_призвище.cpp*).

5. Введіть текст програми.

6. Скомпілюйте програму. Якщо в програмі є помилки, їх необхідно виправити. Якщо помилок немає, то з'явиться повідомлення про успішну компіляції.

7. Запустіть програму на виконання, проаналізуйте результати і переконайтеся в правильності рішення задачі.

8. Виконайте звіт з лабораторної роботи, який повинен містити: титульний аркуш; мета роботи; індивідуальне завдання; алгоритм роботи програми; текст програми; результати роботи програми; висновки.

Зміст звіту

1. Мета лабораторної роботи.
2. Завдання лабораторної роботи.
3. Короткі теоретичні відомості.
4. Текст програми.
5. Алгоритм програми.
6. Результати роботи програми.
7. Висновки.

ЗАВДАННЯ ДЛЯ ІНДИВІДУАЛЬНОЇ РОБОТИ

Варіант № 1.

В одновимірному масиві, що складається з *n* дійсних елементів: замінити всі елементи масиву їх квадратами і впорядкувати елементи масиву по зростанню.

Варіант № 2.

В одновимірному масиві, що складається з *n* дійсних елементів: Замінити всі негативні елементи масиву їх модулями і впорядкувати елементи масиву по спадаючій.

Варіант № 3.

В одновимірному масиві, що складається з *n* дійсних елементів: Перетворити масив таким чином, щоб спочатку розташовувалися всі позитивні елементи, а потім – все негативні (елементи, рівні 0, вважати позитивними).

Варіант № 4.

В одновимірному масиві, що складається з n дійсних елементів: Стиснути масив, видаливши з нього всі елементи, модуль яких не перевищує 1. Вивільнені в кінці масиву елементи заповнити нулями.

Варіант № 5.

В одновимірному масиві, що складається з n дійсних елементів: Стиснути масив, видаливши з нього всі елементи, модуль яких знаходиться в інтервалі $[a, b]$. Вивільнені в кінці масиву елементи заповнити нулями.

Варіант № 6.

В одновимірному масиві, що складається з n дійсних елементів: Перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, рівні нулю, а потім - всі інші.

Варіант № 7.

В одновимірному масиві, що складається з n дійсних елементів: Перетворити масив таким чином, щоб в першій його половині розташовувалися елементи, що стояли в непарних позиціях, а в другій половині – елементи, що стояли в парних позиціях.

Варіант № 8.

В одновимірному масиві, що складається з n дійсних елементів: Перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, модуль яких не перевищує 1, а потім - всі інші.

Варіант № 9.

В одновимірному масиві, що складається з n дійсних елементів: Перетворити масив таким чином, щоб елементи, рівні нулю, розташовувалися після всіх інших.

Варіант № 10.

В одновимірному масиві, що складається з n дійсних елементів: Перетворити масив таким чином, щоб в першій його половині розташовувалися елементи, що стояли в парних позиціях, а в другій половині – елементи, що стояли в непарних позиціях.

Варіант № 11.

В одновимірному масиві, що складається з n дійсних елементів: Стиснути масив, видаливши з нього всі елементи, величина яких знаходиться в інтервалі $[a, b]$. Вивільнені в кінці масиву елементи заповнити нулями.

Варіант № 12.

В одновимірному масиві, що складається з n дійсних елементів: Перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, ціла частина яких лежить в інтервалі $[a, b]$, а потім - всі інші.

Варіант № 13.

В одновимірному. масиві, що складається з n дійсних елементів: Впорядкувати елементи масиву по спадаючій модулів елементів.

Варіант № 14.

В одновимірному масиві, що складається з n дійсних елементів: Впорядкувати елементи масиву по зростанню модулів елементів.

Варіант № 15.

В одновимірному масиві, що складається з n дійсних елементів: Перетворити масив таким чином, щоб спочатку розташовувалися всі негативні елементи, а потім - все позитивні (елементи, рівні 0, вважати позитивними).

Варіант № 16.

В одновимірному масиві, що складається з n дійсних елементів: Замінити всі негативні елементи масиву їх квадратами і впорядкувати елементи масиву по зростанню.

Варіант № 17.

В одновимірному масиві, що складається з n дійсних елементів: Перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, ціла частина яких не перевищує 1, а потім - всі інші.

Варіант № 18.

В одновимірному масиві, що складається з *n* дійсних елементів: Перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, що відрізняються від максимального не більше ніж на 20%, а потім - всі інші.

Варіант № 19.

В одновимірному масиві, що складається з *n* дійсних елементів: Змінити порядок проходження елементів в масиві на зворотний.

Варіант № 20.

В одновимірному масиві, що складається з *n* дійсних елементів: Впорядкувати по зростанню окремо елементи, що стоять на парних місцях, і елементи, що стоять на непарних місцях.

ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

1. Створити одновимірний масив, який складається з чисел Фіббоначі, і вивести його на екран. Відсортувати отриманий масив в порядку спадання і вивести результат на екран.

2. Створити одновимірний масив, який складається з подільників цілого числа *n* заданого з клавіатури. Вивести створений масив на екран. Відсортувати отриманий масив в порядку спадання і вивести результат на екран.

3. Створити одновимірний масив, що складається з цілих тризначних чисел, сума цифр яких дорівнює *n* (ціле число, введене з клавіатури). Вивести створений масив на екран. Відсортувати отриманий масив в порядку спадання і вивести результат на екран.

4. Створити одновимірний масив, що складається з цифр цілого числа *n*, введеного з клавіатури. Вивести створений масив на екран. Відсортувати отриманий масив і вивести результат на екран.

5. Створити одновимірний масив, що складається з цілих тризначних чисел в десяткового запису який немає однакових цифр. Вивести створений масив на екран. Відсортувати отриманий масив в порядку спадання і вивести результат на екран.

6. Створити одновимірний масив, який складається тільки з простих дільників цілого числа *n* заданого з клавіатури. Вивести створений масив

на екран. Відсортувати отриманий масив в порядку спадання і вивести результат на екран.

7. Створити одновимірний масив *a* з 20 цілих чисел. Створити одновимірний масив *b*, який складається з парних елементів масиву a. Відсортувати отриманий масив в порядку зростання і вивести результат на екран.

8. Створити одновимірний масив a з 20 цілих чисел. Створити одновимірний масив *b*, що складається з елементів масиву *a*, в якому необхідно поміняти місцями максимальний і мінімальний елементи. Відсортувати отриманий масив в порядку зростання і вивести результат на екран.

9. Створити одновимірний масив *a* з 20 цілих чисел. Створити одновимірний масив *b*, що складається з зворотного масиву *a*. Вивести на екран масив *b*. Відсортувати отриманий масив в порядку зростання і вивести результат на екран.

10. Створити одновимірний масив *a* з 20 цілих чисел. Створити одновимірний масив *b*, що складається тільки з двозначних чисел масиву *a*. Відсортувати отриманий масив і вивести результат на екран.

11. Створити одновимірний масив *a* з 10 цілих чисел. Перетворити масив таким чином, щоб спочатку розташовувалися всі негативні елементи, а потім – всі позитивні. Вивести на екран створений масив. Відсортувати отриманий масив в порядку зростання і вивести його на екран.

12. Створити одновимірний масив *a* з 20 дійсних чисел. Перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, ціла частина яких не перевищує 1, а потім – всі інші. Вивести на екран створений масив. Відсортувати отриманий масив в порядку зростання і вивести його на екран.

13. Створити одновимірний масив *a* з 20 дійсних чисел. Замінити всі негативні елементи масиву їх квадратами. Впорядкувати отриманий масив в порядку зростання і вивести його на екран.

14. Створити одновимірний масив *a* з 20 дійсних чисел. Створити одновимірний масив *b*, що складається з непарних елементів масиву *a*. Впорядкувати створений масив в порядку спадання.

15. Створити одновимірний масив, що складається з цифр цілого числа n, введеного з клавіатури. Вивести створений масив на екран. Відсортувати отриманий масив і вивести результат на екран.

16. У одновимірному масиві, що складається з n дійсних чисел, впорядкувати по зростанню окремо елементи, що стоять на парних місцях, і елементи, що стоять на непарних місцях.

17. Створити одновимірний масив b, що складається з позитивних елементів одновимірного речового масиву a. Відсортувати отриманий масив по зростанню, результат вивести на екран

18. Є одномірний масив довжиною N=20. Упорядкувати масив методом вибору таким чином, щоб елементи, що знаходяться на парних позиціях розташовувалися за спаданням, а на непарних позиціях – по зростанню.

19. Відсортувати одновимірний масив довжиною N=25 за зростанням методом вибору.

20. Відсортувати одновимірний масив довжиною N 50 за спаданням методом простого обміну.

21. Відсортувати одновимірний масив довжиною N=30 методом простого обміну за зростанням. Після упорядкування в кожній групі повторюваних елементів залишити тільки один, а решта видалити.

Наприклад, якщо відсортований масив має вигляд: 1 2 2 4 4 4 8 9,

кінцевий масив буде мати вигляд: 1 2 4 8 9.

22. Є одномірний масив довжиною N=15. Відсортувати за спаданням методом вибору ті елементи масиву, які є парними числами.

23. Є одномірний масив довжиною N=20. Упорядкувати масив методом простого обміну таким чином, щоб елементи, що знаходяться на парних позиціях розташовувалися по зростанню, а на непарних позиціях – за спаданням.

24. Є одномірний масив довжиною N=20. Відсортувати за зростанням за допомогою методу вибору ті елементи масиву, які розташовуються на непарних позиціях.

25. Є одномірний масив довжиною N=15. Упорядкувати масив методом простого обміну таким чином, щоб елементи, що знаходяться на парних позиціях розташовувалися по зростанню, а на непарних позиціях – за спаданням.

26. Відсортувати за спаданням одновимірний масив довжиною N=20 методом вибору. Після упорядкування в кожній групі повторюваних елементів залишити тільки один, а решта видалити

27. Є двовимірний масив розмірністю NxN, де N = 10. Відсортувати всі стовпці методом простого обміну так, щоб елементи в них розташовувалися за спаданням.

28. Є одномірний масив довжиною N=20. Упорядкувати масив методом простого обміну таким чином, щоб елементи, що знаходяться на парних позиціях розташовувалися за спаданням, а на непарних позиціях – по зростанню.

29. Є одномірний масив довжиною N=20. Упорядкувати масив методом вибору таким чином, щоб елементи, що знаходяться на парних позиціях розташовувалися по зростанню, а на непарних позиціях – за спаданням.

30. Є одномірний масив довжиною N=15. Відсортувати за зростанням за допомогою методу простого обміну ті елементи масиву, які розташовуються на непарних позиціях

Контрольні питання

1. Що таке сортування даних?
2. Які алгоритми сортування відомі?
3. Які алгоритми сортування даних вважаються більш ефективними та чому?
4. У чому полягає алгоритм сортування вибором?
5. У чому полягає алгоритм сортування вставками?
6. У чому полягає алгоритм сортування обміном?
7. У чому полягає алгоритм сортування Шелла?
8. У чому полягає алгоритм сортування злиттям?
9. У чому полягає алгоритм сортування підрахунком?
10. Які характеристики використовуються для порівняння алгоритмів сортування?
11. Яким чином можна поліпшити ефективність алгоритму сортування обміном?

СПИСОК ЛІТЕРАТУРИ

1. Васильченков О.Г., Тверитникова О Є., Крилова В. А. Основы алгоритмизации и программирования на С++. Учебное пособие. Харьков: НТУ "ХПІ", 2015. 228 с.

2. Тверитникова О Є., Крылова В. А. Методические указания по информатике «Основы программирования на С++». Часть 1. Харьков: НТУ “ХПІ”, 2013. 52 с.

3. Тверитникова О Є., Крылова В. А., Опрышкина М.И Методические указания по информатике «Основы программирования на С++». Часть 2. Харьков: НТУ “ХПІ”, 2014. 68 с.

4. Тверитникова О Є., Крылова В. А. Методические указания по информатике «Основы программирования на С++». Часть 3. Двухмерные массивы. Харьков: ТУ «ХПІ», 2017. 52 с.

5. Ткачук В.М. Програмування на С++. Лабораторний практикум Івано-Франківськ: Видавництво Прикарпатського національного університету імені Василя Стефаника, 2011. 160 с.

6. Жуковський С.С., Вакалюк Т.А. Програмування мовою С++. Структурне програмування (лабораторний практикум). Навчальний посібник для студентів фізико-математичного факультету. Житомир: Вид-во ЖДУ, 2011. 92 с.

7. Віник В.Ю. Алгоритмічні мови та основи програмування: мова С. навчальний посібник. Житомір: ЖДТУ, 2007. 328 с.

8. Методичні вказівки до лабораторних робіт з курсів «Інформатика», розділ «Персональні комп’ютери та основи програмування», частина І / І.П. Голубєва та ін. Київ: НТУУ «КПІ», 2013. 124 с.

9. Страуструп Бьерн. Язык программирования С++. Москва: Бином, 2002. 1098 с.

10. Льовкін В.М. Методичні вказівки до виконання лабораторних робіт з дисципліни “Основи програмування” для студентів спеціальності 121 “Інженерія програмного забезпечення” (всіх форм навчання). Запоріжжя: ЗНТУ, 2016. 64 с.

11. Шилдт Г. Теория и практика С++. Санкт-Петербург: «BHV-Санкт-Петербург», 1996. 416 с.

12. Павловская Т.А., Щупак Ю.А. Структурное программирование С/С++. Практикум. Санкт-Петербург: Питер, 2007. 239 с.

Навчальне видання

ТВЕРИТНИКОВА Олена Євгенівна
КРИЛОВА Вікторія Анатоліївна

ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ С++. ВИКОРИСТАННЯ АЛГОРИТМІВ СОРТУВАННЯ

Методичні вказівки

до виконання практичних та лабораторних робіт
для студентів спеціальностей
152 «Метрологія та вимірювальна техніка»
151 «Автоматизація та комп'ютерно-інтегровані технології»
172 «Телекомунікація та радіотехніка»

Відповідальний за випуск Качанов П.О.

Роботу до друку рекомендував О.В. Дудник

В авторській редакції

План 2020 р., поз. 60

Підп. до друку 03.04.20. Формат 60×84 1/16. Папір офсетний.
Riso-друк. Гарнітура Times New Roman. Ум. друк. арк. 1,5.
Наклад 50 прим. Зам. № _____. Ціна договірна.

Видавець Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2

Виготовлювач_____

_____