**Форма № Н-6.01у.1**

Кременчуцький національний університет імені Михайла Остроградського

(повне найменування вищого навчального закладу)

\_\_\_\_ Кафедра інформаційно–управляючих систем \_

(повна назва кафедри, циклової комісії)

# **КУРСОВИЙ ПРОЕКТ**

**(РОБОТА)**

з дисципліни «Теорія алгоритмів»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва дисципліни)

на тему: «Алгоритм сортування гнома»\_\_\_\_

Студента \_2\_ курсу КН–17–1 групи

ступінь вищої освіти \_\_\_\_«Бакалавр»\_\_\_\_\_\_\_\_\_

(бакалавр, магістр)

спеціальність 122 – «Комп’ютерні науки»»\_

освітньо–професійна програма «Інформаційні\_ управляючі системи та технології»\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ляпко Б.І.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник професор кафедри ІУС, д. т. н., доц.\_

Притчин С. Е.\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_. Оцінка: ЄКTС \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_С. Е. Притчин\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_А. П. Оксанич\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Н. В. Рилова \_\_

(підпис) (прізвище та ініціали)

м. Кременчук 2018 рік

**Форма № Н-9.01у.2**

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Кафедра \_ інформаційно–управляючих систем ­­­­

Дисципліна «Теорія алгоритмів» ­­

Освітній ступень «Бакалавр»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність 122 – «Комп’ютерні науки»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Освітньо–професійна програма «Інформаційні\_ управляючі системи та технології»­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Курс 2 група КН–17–1 семестр \_3\_

ЗАВДАННЯ

на курсовий проект (роботу) студенту

Ляпко Богдану Ігоровичу

(прізвище, ім’я, по батькові)

1. Тема проекту: Алгоритм сортування гнома \_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Термін здачі студентом проекту: 10 грудня 2018 р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до проекту: консольний інтерфейс, введення розміру масиву,

відображення результатів алгоритму алгоритму, внутрішнє та зовнішнє сортування, середовище розробки\_Visual Studio 2017, мова програмування С++\_\_\_\_

4. Зміст розрахунково–пояснювальної роботи (перелік питань, що підлягають\_ розробці): 1) Приклад використання крок за кроком; 2) Опис методів програми.

5. Перелік графічного матеріалу: блок–схема алгоритму програми.\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Дата видачі завдання: 21 вересня 2018 р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  п/п | Назва етапів курсового проекту | Терміни виконання етапів проекту | Вказівки та  зауваження викладача (з зазначенням дати консультації) | Оцінювання етапів проекту | | |
| за націо– нальною шкалою | за шкалою ЄКТС | кількість балів |
| 1 | Етап 1  Аналіз предметної області | 10.10.18–  11.10.18 |  |  |  |  |
| 2 | Етап 2  Розробка алгоритму програми | 12.10.18–  19.10.18 |  |  |  |  |
| 3 | Етап 3  Розробка інтерфейсу користувача | 20.10.18–  26.10.18 |  |  |  |  |
| 4 | Етап 4  Розробка керівництва користувача | 27.10.18–  11.11.18 |  |  |  |  |
| 5 | Етап 5  Оформлення пояснювальної записки | 12.11.18–  30.11.18 |  |  |  |  |
| 6 | Етап 9  Захист | 10.12.18 |  |  |  |  |
|  | Разом | 9 тижнів |  |  |  |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_С. Е. Притчин\_\_\_

(підпис) (ініціали та прізвище)

«21» \_\_09\_\_ 2018 р.

РЕФЕРАТ

Курсова робота містить: 23 с., 2 розділи, 2 рисунки, 5 використаних джерел.

Об’єкт дослідження – алгоритм сортування гнома.

Метою проекту є створення додатку, який реалізує алгоритм сортування гнома.

Для досягнення мети в проекті виконуються такі задачі як: детальний аналіз загального підходу до розробки алгоритму, вивчення тонкощів і нюансів конкретного алгоритму сортування, вибір найбільш підходящих структур даних, для якісної реалізації алгоритму сортування.

Результатом виконання всіх етапів є додаток, створений у середовищі програмування .Net на мові C++, який реалізує алгоритм парного та непарного сортування.

АЛГОРИТМ СОРТУВАННЯ, СОРТУВАННЯ ГНОМА, МОВА ПРОГРАММУВАННЯ С++

**ЗМІСТ**

*Змн*.

Арк.

№ докум.

*Підпис*

Дата

Арк.

1

*122 – КР.2018.01.000 ПЗ*

*Розроб.*

*Ляпко Б.І.*

*Перевір.*

Притчин С. Е.

*Н. Контр.*

Рилова Н. В.

*Затверд.*

*Алгоритм сортування гнома*

*Пояснювальна записка*

Літ.

Аркушів

31

КрНУ Кафедра ІУС

ВСТУП......................................................................................................................2

**1** АНАЛІТИЧНИЙ РОЗДІЛ………………………………………………..………4

1.1Аналіз предметної області………………………………………..........4

1.2 Аналіз загального методу проектування алгоритму………………....7

1.3 Аналіз алгоритму сортування гнома……………..………………...…13

Висновки до розділу……………………………………………………………….16

2 ПРОГРАМНИЙ РОЗДІЛ………………………………………………………...17

2.1 Приклад використання крок за кроком ………………………………………………………………………………17

2.2 Опис коду програми….…………………………..…..………………..17

Висновки до розділу ………………………………………………………………22

ВИСНОВКИ………………………………………………………………………23

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ……………………………………….24

ДОДАТОК А………………………………………………………………………25

**ВСТУП**

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Для рішення задач треба знати, що дано, і що варто одержати. Іншими словами, задача являє собою сукупність двох об’єктів: вихідних даних і шуканих результатів. Щоб одержати результати, необхідно знати метод рішення задачі, тобто мати у своєму розпорядженні розпо­рядження (інструкцію, правило), у якому зазначено, які дії й у якому порядку варто виконати, щоб вирішити задачу (одержати шукані результати). Розпорядження, що визначає порядок виконання дій над даними з метою одержання шуканих результатів, називається алгоритмом.

Об’єктом курсової роботи є алгоритм сортування гнома, один із найпростіших алгоритмів сортування (на думку багатьох — найпростіший). Ім'я походить від голландського садового гнома, якого ставлять між квітковими горщиками. Якщо два сусідні від гнома горщики йдуть у правильному порядку, гном йде на одну позицію вперед. Якщо ж вони у неправильному порядку - міняє ці два горщики місцями і йде на одну позицію назад (щоб знову перевірити порядок).

Алгоритм концептуально простий, і не потребує навіть вкладених циклів. Його швидкодія рівна O(n^{2}), однак, на практиці, може працювати й швидше у режимі сортування вставками.

**1 АНАЛІТИЧНИЙ РОЗДІЛ**

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

1.1Аналіз предметної області

Теорія алгоритмів – це наука, що вивчає загальні властивості та закономір­ності алгоритмів, різноманітні формальні моделі їх подання. На основі формалі­зації поняття алгоритму можливе порівняння алгоритмів за їх ефективністю, пере­вірка їх еквівалентності, визначення областей застосовності [1].

Алгоритм – це точна кінцева система правил, що визначає зміст і порядок дій виконавця над деякими об’єктами (вихідними і проміжними даними) для одер­жання (після кінцевого числа кроків) шуканого результату.[2]

Спочатку під алгоритмами розуміли тільки правила виконання чотирьох арифме­тичних дій над десятковими числами. Надалі це поняття стали використовувати для позначення будь–якої послідовності дій, що приводить до рішення поставленої задачі.

Інтуїтивне поняття алгоритму включає в себе декілька загальних рис, які часто визнаються характерними для поняття алгоритму.

1. Алгоритм – це процес послідовної побудови величин, який проходить в дис­кретному часі таким чином, що в початковий момент задається вихідна скінчена множина величин, а в кожний наступний система величин знаходиться за цілком визначеним законом (програмою) із системи величин, які були в попередній мо­мент часу (дискретність алгоритму).

2.  Система величин, що одержується в якийсь відмінний від початкового момент часу, однозначно визначається системою величин, одержаних в попередні мо­менти часу (детермінованість алгоритму).

3. Закон одержання подальших систем величин із попередніх повинен бути про­стим і локальним (елементарність кроків алгоритму).

4. Якщо спосіб одержання наступної величини із якої–небудь заданої величини не дає результату, то слід вказати, що вважати результатом алгоритму (направленість алгоритму).

5.  Початкова система величин може вибиратись із деякої потенціальної нескін­ченної множини (масовість алгоритму).

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Поняття алгоритму, в деякій мірі визначене пунктами 1) – 5), звичайно, не строге: в формулювання цих пунктів зустрічаються слова спосіб, величина, простий, локальний, точний зміст яких не встановлено. Надалі це нестроге поняття алгоритму буде називатися безпосереднім або інтуїтивним поняттям алгоритму.

Користуючись інтуїтивним поняттям алгоритму, можна описувати процес розв’язку тієї чи іншої задачі, але з його допомогою не можна упевнитися в тому, що описаний процес являє собою алгоритм.

Способи представлення алгоритму

1.  Словесні.

2.  Словесно–формульні.

3.  Графічні.

4.  Скінчений набір кодів.

При складанні алгоритмів можна поєднувати різні форми подання алгоритмів.

Процес алгоритмізації – це визначення елементарних дій та порядку їх виконання для розв’язання поставленого завдання. Існують різні способи запису алгоритмів (словесний, формульно-словесний, метод блок-схем, програмний та ін.), які застосо­вуються для представлення алгоритму у вигляді, що однозначно розуміється і розробни­ком, і виконавцем алгоритму.

Для опису алгоритмів людина часто користується природною мовою, але для запису багатьох алгоритмів природна мова виявилась незручною, тому виникла необхідність у створенні штучних мов, наприклад мови математичних формул, хімічних процесів тощо. Найбільше поширення для запису логічної структури алгоритмів отримали графічні (структурні) схеми, які спрощують складання та аналіз алгоритму, полегшують перехід від запису алгоритму до написання програми.

Графічна схема (блок–схема) алгоритму — це графічне зображення алгоритму у вигляді спеціальних блоків з необхідними словесними поясненнями. Кожний етап алгоритму представляється у вигляді геометричної фігури (блоку), що має певну форму в залежності від характеру операції. Блоки на схемі з’єднуються стрілками (лініями зв’язку), які визначають послідовність виконання операцій та утворюють логічну структуру алгоритму.

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Важливою особливістю базових структур алгоритмів є те, що вони мають

один вхід і один вихід, що дозволяє при відносній незалежності конструювати

окремі блоки алгоритмів, а потім окремо розроблені структури з’єднувати між

собою (вихід однієї базової структури сполучається із входом іншої). Весь

алгоритм представляє лінійну послідовність базових структур

1.2 Аналіз загального методу проектування алгоритму

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Сортуванням називають впорядковування по ключах (тобто за якою-

небудь ознакою) елементів деякої структури даних на якій визначено відношення порядку.

У залежності від того, чи знаходяться елементи структур даних у внутрішній (оперативній) пам’яті або у зовнішній пам’яті (на зовнішніх пристроях), розрізняють внутрішнє та зовнішнє сортування.

Нехай існують послідовність a0, a1... an і функція порівняння, яка на будь–яких двох елементах послідовності приймає одне з трьох значень: менше, більше або дорівнює. Задача сортування полягає у перестановці елементів послідовності так, щоб виконувалася умова: ai <= ai+1, для всіх i від 0 до n.

Якщо значення функції порівняння залежить тільки від поля x, то x називають ключем, по якому проводиться сортування. На практиці, в якості x часто виступає число, а поле у зберігає будь–які дані, жодним чином не впливаючи на роботу алгоритму.

Мабуть, жодна інша проблема не породила такої кількості найрізноманітніших рішень, як задача сортування. Чи існує певний „універсальний, найкращий алгоритм“? Маючи приблизні характеристики вхідних даних, можна підібрати метод, який працює оптимальним чином.

Для того, щоб обґрунтовано зробити такий вибір, розглянемо параметри,

по яких проводититься оцінка алгоритмів.

1. Час сортування – основний параметр, що характеризує швидкодію алгоритму.

2. Пам’ять– ряд алгоритмів вимагає виділення додаткової пам’яті під тимча-сове зберігання даних. При оцінці пам’яті, що використовується, не враховує-ться місце, яке займає початковий масив і незалежні від вхідної послідовності витрати, наприклад, на зберігання коду програми.

3. Стійкість– стійке сортування не міняє взаємного розташування рівних елементів. Така властивість може бути дуже корисною, якщо вони складаються з декількох полів, як на рис. 5, а сортування відбувається по одному з них, наприклад, по x.

4. *Природність поведінки*– ефективність методу при обробці вже відсортованих, або частково відсортованих даних. Алгоритм поводиться природно, якщо враховує цю характеристику вхідної послідовності і працює краще.

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Ще однією важливою властивістю алгоритму є його сфера застосування. Тут основних позицій дві:

-внутрішні сортування працюють з даним в оперативній пам’яті з довільним доступом;

- зовнішні сортування упорядковують інформацію, розташовану на зовнішніх носіях.

Це накладає деякі додаткові обмеження на алгоритм: доступ до носія здійснюється послідовним чином: в кожний момент часу можна зчитати або записати тільки елемент, наступний за слідуючим; o об’єм даних не дозволяє їм розміститися в ОЗП; Крім того, доступ до даних на носію здійснюється набагато повільніше, ніж операції з оперативною пам’яттю.

Даний клас алгоритмів ділиться на два основні підкласи:

· Внутрішнє сортуванняоперує з масивами, що цілком поміщаються в оперативній пам’яті з довільним доступом до будь–якої комірки. Дані звичайно сортуються на тому ж місці, без додаткових витрат.

· Зовнішнє сортуванняоперує з пристроями великого об’єму, що запам’ятовують, але з доступом не довільним, а послідовним (сортування файлів), тобто у даний момент ми “бачимо” тільки один елемент, а витрати на перемотування у порівнянні з пам’яттю невиправдано великі . Це приводить до спеціальних методів сортування, які звичайно використовують додатковий дисковий простір.

Функція складності алгоритму

 Для вирішення багатьох проблем існує безліч різних алгоритмів. Який з них вибрати для вирішення конкретного завдання? Це питання дуже ретельно опрацьовується в програмуванні. Ефективність програми є дуже важливою її характеристикою.

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Ефективність програми має дві складові: пам’ять (або простір) і час. Просторова ефективність вимірюється кількістю пам’яті, потрібної для виконання програми. Комп’ютери володіють обмеженим об’ємом пам’яті. Якщо дві програми реалізують ідентичні функції, то та, яка використовує менший об’єм пам’яті, характеризується більшою просторовою ефективністю. Інколи пам’ять стає домінуючим чинником в оцінці ефективності програм. Проте останніми роками у зв’язку з швидким її здешевленням ця складова ефективності поступово втрачає своє значення. Тимчасова ефективність програми визначається часом, необхідним для її виконання. Кращий спосіб порівняння ефективність алгоритмів полягає в зіставленні їх порядків складності. Цей метод застосовний як до тимчасової, так і просторовій складності. Порядок складності алгоритму виражає його ефективність зазвичай через кількість оброблюваних даних. Наприклад, деякий алгоритм може істотно залежати від розміру оброблюваного масиву. Якщо, скажімо, час обробки подвоюється з подвоєнням розміру масиву, то порядок тимчасової складності алгоритму визначається як розмір масиву.

Порядок алгоритму – це функція, домінуюча над точним вираженням тимчасовій складності. Функція f(n) має порядок *0(g(n)),*якщо є константа *К*і лічильник *n0*такі, що *f(n)(K\*g(n))*для *п*> *n0.* Наприклад: відомо, що точний час обробки масиву визначається з рівняння:

Види функції складності алгоритмів

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Для оцінки ефективності алгоритмів використовується функція складності алгоритму, яка позначається прописною буквою О, в круглих дужках записується аргумент. Наприклад, функція складності 0(n^2) читається як функція складності порядка n^2. функція складності алгоритму — це функція, що визначає кількість порівнянь, перестановок, а також тимчасові і ресурсні витрати на реалізацію алгоритму.

Функція складності приймає наступний ряд значень (Рис. 1.1).

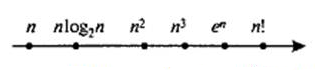


Рис. 1.1- Функція складності

Чим правіше на осі розташована функція складності, тим менш ефективний алгоритм.

· Методи проста вставка, простий і стандартний обмін мають функцію складності *О(п^2).*

· Метод Хоара і пірамідальне сортування мають функцію складності 0(n*log2n)*

· Метод Шелла і турнірне сортування мають функцію складності 0(0,3n(1оg2n)^2).

**Cкладність алгоритмів сортування**

N)

0(1)

У алгоритмах константної складності більшість опера­цій| в програмі виконуються один або кілька разів. Будь–який алгоритм, що завжди вимагає незалежно від розміру даних од­ного і того ж часу, має константну складність.

Час роботи програми лінійний, зазвичай звично коли кожен елемент вхідних даних потрібно обробити лише лінійне число разів.

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

0(N2), 0(N3), 0(Na)

Поліноміальна складність*. 0(N2) —*квадратична складність, *0(N3)*— кубічна складність

0(Log|(N))

Коли час роботи програми логарифмічний, програма зачинає працювати набагато повільніше із збільшенням *N.*Такий час роботи зустрічається зазвичай в програмах, які ділять велику проблему на маленьких і вирішують їх окремо.

0(N\*log|(N))

Такий час роботи мають ті алгоритми, які ділять велику проблему намаленьких, а потім, вирішивши їх, сполучають з’єднують їх рішення|вирішення|.

0(2N)

Експоненціальна складність*.*Такі алгоритми найчастіше виникають врезультаті підходу, що іменується методом грубої сили.

Впорядкування елементів безлічі множини в зростаючому або спадаючому порядку(ладі) називається сортуванням.

З (із) впорядкованими елементами простіше працювати, чим з (із) хаотично розташованими (схильними): легко знайти необхідні елементи, виключити, вставити нові. Сортування застосовується при транс­ляції| програм, приорганізації наборів даних на зовнішніх носіях, при створенні|створінні|бібліотек, каталогів, баз даних і так далі

Алгоритми сортування можна розбити на наступні групи (мал. 2.1). Зазвичай сортовані елементи безлічі називають записами і позначають через *k1, k2*..., *кn.*

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*



Рис. 1.2 – Методи сотрування

1.3 Аналіз алгоритму сортування гнома.

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Сортування гнома вперше була запропонована 2 жовтня 2000 року Хамідом Сарбази-Азад (Hamid Sarbazi-Azad). Він назвав її «Дурне сортування, найпростіший алгоритм сортування лише з одним циклом ...». І дійсно, дурний цей метод чи ні, але в ньому задіяний, ніяк в більшості угруповань - два або більше циклів, а тільки один. Пізніше, голландський вчений Дік Грун, на сторінках однієї зі своїх книг, привів для гномів сортування наступну аналогію: «Сортування гнома заснована на техніці, використовуваної звичайним голландським садовим гномом. Ось як садовий гном сортує ряд квіткових горщиків. По суті, він дивиться на два сусідніх квіткових горщика, якщо вони знаходяться в правильному порядку, то він переходить на один горщик вперед, інакше він змінює їх місцями і повертається на один горщик назад. Граничні умови: якщо позаду немає жодного горщика - він крокує вперед, а якщо немає такого горщика, тоді він закінчив». Так описав основну ідею алгоритму гномів сортування Дік Грун. Замінимо гнома і горщики на більш формальні об'єкти, наприклад на покажчик (номер поточного елемента) і елементи масиву, і розглянемо принцип роботи алгоритму ще раз. Спочатку покажчик ставитися на 2-ий елемент масиву (в C ++ він має номер 1, а в Паскалі номер 2). Потім відбувається порівняння двох сусідніх елементів A [i] і A [i-1], т. Е. Порівнюються перший елемент (i-1) і другий (i). Якщо ті стоять на своїх позиціях, то зрушуємо покажчик на позицію i + 1 і продовжуємо порівняння з нового положення; інакше міняємо елементи місцями, і, оскільки в якийсь момент може виявитися, що елементи в лівому підмасиві стоять не на своїх місцях, зрушуємо покажчик на одну позицію вліво, здійснюючи таке порівняння з нової позиції. Так алгоритм залишиться активним до тих пір, поки весь масив не буде впорядковано. Тут слід виділити два важливих моменти. По-перше, процес упорядкування закінчується, тоді коли не виконується умова і<n.Де n – розмір масива.   
По-друге, як було сказано, покажчик переміщається як вперед по списку, так і назад, і в тому випадку якщо він виявиться над першим елементом, його відразу ж слід змістити на одну позицію вправо, т. д. В іншому випадку доведеться порівнювати два елемента, одного з яких не існує.

Висновки до розділу

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Під час виконання аналітичного розділу було виконано глибоке дослідження предметної області, аналіз загального методу проектування алгоритму,атакож аналіз алгоритму сортуванняч бінарним деревом. Для виконання цих задач, було виконано спеціальну літературу, Інтернет ресурси, а також інформація отримана підчас вивчення курсу«Теорія алгоритмів».

В результаті встановлення етапів, було отримано загальні відомості про предметну область, інформацію про такий до конструювання алгоритмів, як метод зменшення розміру задачі, а також виявлено нюанси роботи алгоритму сортуванняч бінарним деревом,його переваги та недоліки.

**2 ПРОГРАМНИЙ РОЗДІЛ**

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

2.1 Приклад використання крок за кроком.

Відсортуємо масив числових елементів [4] [2] [7] [3] від найбільшого до найменшого:

[4] [2] [7] [3] (ініціалізація. i = 1, а j = 2.)

[4] [2] [7] [3] (нічого не робити, однак, тепер i = 2, а j = 3.)

[4] [7] [2] [3] (міняємо місцями a[2] та a[1]. однак, тепер i = 1, а j й досі = 3.)

[7] [4] [2] [3] (міняємо місцями a[1] and a[0].однак, тепер i = 1,а j й досі= 3.)

[7] [4] [2] [3] (нічого не робити, однак, тепер i = 3, а j = 4.)

[7] [4] [3] [2] (міняємо місцями a[3] and a[2]. однак, тепер i = 2, а j = 4.)

[7] [4] [3] [2] (нічого не робити, однак, тепер i = 4, а j = 5.)

на цьому місці цикл завершується, оскільки і не < 4.

2.2 Опис коду програми.

#include "stdafx.h"

#include <iostream>

using namespace std;

int n;

void Gnome\_sort(int i, int j, int \*mas)

{

while(i<n)

{

if(mas[i-1]<=mas[i]){i=j;j++;}

else

{

int t=mas[i];

mas[i]=mas[i-1];mas[i-1]=t;

i--

if (i= =0){i=j;j++;}

}

cout<<;//Отсортированный масив.

for(i=0;i<n;i++)//вывод массива.

cout<<mas[i]<<” “;

}

void main()

}

setlocale(LC\_ALL,”Rus”);

int m,k;

cout<<”Размер массива>”;cin>>n;

int\*a=new int [n];

for (k=0;k<n;k++)//ввод массива

{cout<<k+1<<”элемент>”;cin>>a[k];}

k=1;m=2;

Gnome\_sort (k,m,a);//вызов функции сортировки

delete []a;

system(“pause>>void”);

}

Висновки до розділу

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Наведений Грунем код дозволяє впорядкувати масив, але все ж він може бути трохи поліпшений. Оптимізація потребують невеликого редагування, зокрема, додавання однієї допоміжної змінної, що дозволить позбутися від зайвих порівнянь.

**ВИСНОВКИ**

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Метою курсової роботи було створення додатку, який реалізує виконання алгоритму гнома , а також має зрозумілий інтерфейс.

На початку роботи виконано аналіз предметної області. Потім був проаналізований метод проектування алгоритму задля чіткого розуміння використовуваного підходу, після чого вивчено сам алгоритм сортування гнома.

Для реалізації проекту було обрано C++ ­­­– мову програмування високого рівня з підтримкою кількох парадигм програмування.

Було описано структуру методів, які стосуються реалізації алгоритму.

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

1. <https://studfiles.net/preview/1755574/>

2. [https://www.ua5.org/osnprog/419–algoritmi–ta–yix–vlastivosti.html](https://www.ua5.org/osnprog/419-algoritmi-ta-yix-vlastivosti.html)

3. <http://rosettacode.org/wiki/Sorting_algorithms/Gnome_sort#Common_Lisp>

4. <https://uk.wikipedia.org>

5. https://intellect.ml/gnomya-sortirovka-4381

ДОДАТОК В

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*122 – КР.2018.01.000 ПЗ*

Блок–схема алгоритму

