**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 1 з дисципліни

«Проектування алгоритмів»

„ **Проектування і аналіз алгоритмів зовнішнього сортування**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-11 Прищепа Владислав Станіславович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

Київ 2022

Зміст

[1 Мета лабораторної роботи 3](file:///C:\Users\User\Downloads\Telegram%20Desktop\lr1_2022.ukr.docx#_Toc109342184)

[2 ЗаВдання 4](file:///C:\Users\User\Downloads\Telegram%20Desktop\lr1_2022.ukr.docx#_Toc109342185)

[3 Виконання 6](file:///C:\Users\User\Downloads\Telegram%20Desktop\lr1_2022.ukr.docx#_Toc109342186)

[3.1 Псевдокод алгоритму 6](file:///C:\Users\User\Downloads\Telegram%20Desktop\lr1_2022.ukr.docx#_Toc109342187)

[3.2 Програмна реалізація алгоритму 6](file:///C:\Users\User\Downloads\Telegram%20Desktop\lr1_2022.ukr.docx#_Toc109342188)

[3.2.1 Вихідний код 6](file:///C:\Users\User\Downloads\Telegram%20Desktop\lr1_2022.ukr.docx#_Toc109342189)

[Висновок 7](file:///C:\Users\User\Downloads\Telegram%20Desktop\lr1_2022.ukr.docx#_Toc109342190)

[Критерії оцінювання 8](file:///C:\Users\User\Downloads\Telegram%20Desktop\lr1_2022.ukr.docx#_Toc109342191)

# Мета лабораторної роботи

Мета роботи – вивчити основні алгоритми зовнішнього сортування та способи їх модифікації, оцінити поріг їх ефективності.

# Завдання

Згідно варіанту (таблиця 2.1), розробити та записати алгоритм зовнішнього сортування за допомогою псевдокоду (чи іншого способу за вибором).

Виконати програмну реалізацію алгоритму на будь-якій мові програмування та відсортувати випадковим чином згенерований масив цілих чисел, що зберігається у файлі (розмір файлу має бути не менше 10 Мб, можна значно більше).

Здійснити модифікацію програми і відсортувати випадковим чином згенерований масив цілих чисел, що зберігається у файлі розміром не менше ніж двократний обсяг ОП вашого ПК. Досягти швидкості сортування з розрахунку 1Гб на 3хв. або менше.

Рекомендується попередньо впорядкувати серії елементів довжиною, що займає не менше 100Мб або використати інші підходи для пришвидшення процесу сортування.

Зробити узагальнений висновок з лабораторної роботи, у якому порівняти базову та модифіковану програми. У висновку деталізувати, які саме модифікації було виконано і який ефект вони дали.

Таблиця 2.1 – Варіанти алгоритмів

|  |  |
| --- | --- |
| **№** | **Алгоритм сортування** |
| 1 | Пряме злиття |
| 2 | Природне (адаптивне) злиття |
| 3 | Збалансоване багатошляхове злиття |
| 4 | Багатофазне сортування |
| 5 | Пряме злиття |
| 6 | Природне (адаптивне) злиття |
| 7 | Збалансоване багатошляхове злиття |
| 8 | Багатофазне сортування |
| 9 | Пряме злиття |
| 10 | Природне (адаптивне) злиття |
| 11 | Збалансоване багатошляхове злиття |
| 12 | Багатофазне сортування |
| 13 | Пряме злиття |
| 14 | Природне (адаптивне) злиття |
| 15 | Збалансоване багатошляхове злиття |
| 16 | Багатофазне сортування |
| 17 | Пряме злиття |
| 18 | Природне (адаптивне) злиття |
| 19 | Збалансоване багатошляхове злиття |
| 20 | Багатофазне сортування |
| 21 | Пряме злиття |
| 22 | Природне (адаптивне) злиття |
| 23 | Збалансоване багатошляхове злиття |
| 24 | Багатофазне сортування |
| 25 | Пряме злиття |
| 26 | Природне (адаптивне) злиття |
| 27 | Збалансоване багатошляхове злиття |
| 28 | Багатофазне сортування |
| 29 | Пряме злиття |
| 30 | Природне (адаптивне) злиття |
| 31 | Збалансоване багатошляхове злиття |
| 32 | Багатофазне сортування |
| 33 | Пряме злиття |
| 34 | Природне (адаптивне) злиття |
| 35 | Збалансоване багатошляхове злиття |

# Виконання

## Псевдокод алгоритму

### Не модифікований

function coun(quan):

ah = log2(quan)

quamer = int(ah)

if (fmod(ah, 1) > 0.0):

quamer = quamer + 1

return quamer

function merge(Name, quan):

Help1 = "help1.txt"

Help2 = "help2.txt"

ammer = coun(quan)

for i in range 0, ammer whith step 1:

outp.read(Name)

fill1.write(Help1)

fill2.write(Help2)

hf0 = false

hf1 = false

h = pow(2, i)

for j in range 0, quan with step 1:

getline(outp, lh)

if (j / h) % 2 == 0:

if hf0 is true:

fill1 << "\n"

else:

hf0 = true

fill1 << lh

else:

if hf1 is true:

fill2 << "\n"

else:

hf1 = true

fill2 << lh

outp.close()

fill1.close()

fill2.close()

refill.write(Name)

outp1.read(Help1)

outp2.read(Help2)

flag1 = true

flag2 = true

if outp1.eof() is false:

getline(outp1, lh1)

else:

flag1 = false

if outp2.eof() is false:

getline(outp2, lh2)

else

flag2 = false

hf0 = false

while flag1 or flag2 is true:

a = 0

b = 0

while (a < h or b < h):

if flag1 and flag2 is true:

if a >= h:

if hf0 is true:

refill << "\n"

else:

hf0 = true

refill << lh2

b = b + 1

if outp2.eof() is false:

getline(outp2, lh2)

else:

flag2 = false

else if b >= h:

if hf0 is true:

refill << "\n”

else:

hf0 = true

refill << lh1

a = a + 1

if outp1.eof() is false:

getline(outp1, lh1)

else:

flag1 = false

else:

if stoi(lh1) <= stoi(lh2):

if hf0 is true:

refill << "\n

else:

hf0 = true

refill << lh

a = a + 1

if outp1.eof() is false:

getline(outp1, lh1)

else:

flag1 = false

else:

if hf0 is true:

refill << "\n"

else:

hf0 = true

refill << lh2

b = b + 1

if outp2.eof() is false:

getline(outp2, lh2)

else:

flag2 = false

else:

if flag1 is true:

if hf0 is true:

refill << "\n"

else:

hf0 = true

refill << lh1

if outp1.eof() is false:

getline(outp1, lh1)

else:

flag1 = false

else:

if hf0 is true:

refill << "\n"

else:

hf0 = true

refill << lh2

if outp2.eof() is false:

getline(outp2, lh2)

else:

flag2 = false

if flag1 and !flag2 is false:

a = h

b = h

outp1.close()

outp2.close()

refill.close()

### Модифікований

function merge(arr, p, q, r):

n1 = q - p + 1

n2 = r - q

L[n1], M[n2]

for i in range 0, n1 with step 1:

L[i] = arr[p + i]

for j in range 0, n2 with step 1:

M[j] = arr[q + 1 + j]

i = 0

j = 0

k = p

while i < n1 and j < n2:

if L[i] <= M[j]:

arr[k] = L[i]

i = i + 1

else:

arr[k] = M[j]

j = j + 1

k = k + 1

while i < n1:

arr[k] = L[i]

i = i + 1

k = k + 1

while j < n2:

arr[k] = M[j]

j = j + 1

k = k + 1

function mergeSort(arr, l, r):

if l < r:

m = l + (r - l) / 2

mergeSort(arr, l, m)

mergeSort(arr, m + 1, r)

merge(arr, l, m, r)

function GenSort(siz, am, Name):

Sorted = "sorted.txt"

Help = "help.txt"

SortedP.write(Sorted)

SortedP.close()

Readf.read(Name)

line = ""

for i in range 0, am with step 1:

hf0 = false

arr[siz]

for j in range 0, siz with step 1:

getline(Readf, line)

arr[j] = stoi(line)

mergeSort(arr, 0, siz - 1)

line = ""

SortedY.read(Sorted)

HelpP.write(Help)

for j in range 0, i \* siz with step 1:

getline(SortedY, line)

if hf0 is true:

HelpP << "\n”

else:

hf0 = true

HelpP << line

SortedY.close()

HelpP.close()

hf0 = false

SortedN.write(Sorted)

HelpN.read(Help)

c = 0

q = 0

getline(HelpN, line)

while c < siz or q < siz \* i:

if c < siz and q < siz \* i:

if arr[c] <= stoi(line):

if hf0 is true:

SortedN << "\n"

else:

hf0 = true

SortedN << arr[c]

c = c + 1

else:

if hf0 is true:

SortedN << "\n"

else:

hf0 = true

SortedN << stoi(line)

if q < siz \* i:

getline(HelpN, line)

q = q + 1

else:

if c < siz:

if hf0 is true:

SortedN << "\n"

else:

hf0 = true

SortedN << arr[c]

c = c + 1

else:

if hf0 is true:

SortedN << "\n”

else:

hf0 = true

SortedN << stoi(line)

if q < siz \* i:

getline(HelpN, line)

q = q + 1

delete[] arr

SortedN.close()

HelpN.close()

}

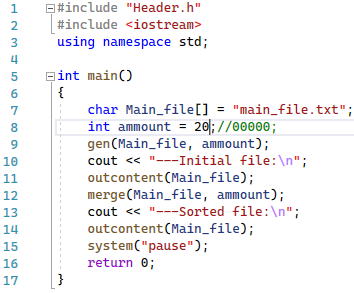
Readf.close()

}

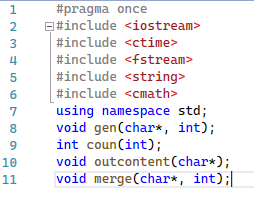
## Програмна реалізація алгоритму

### Вихідний код

AP\_laba\_1\_classic.cpp:

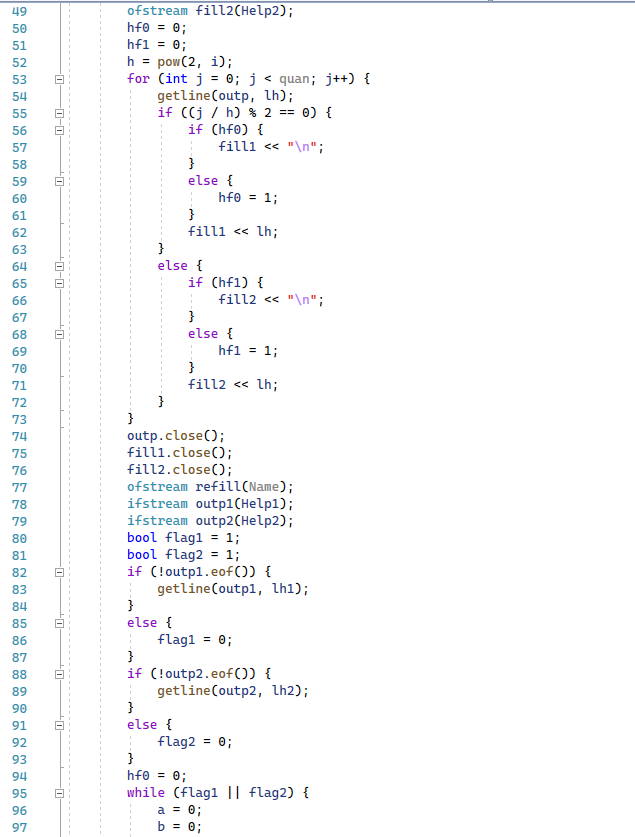


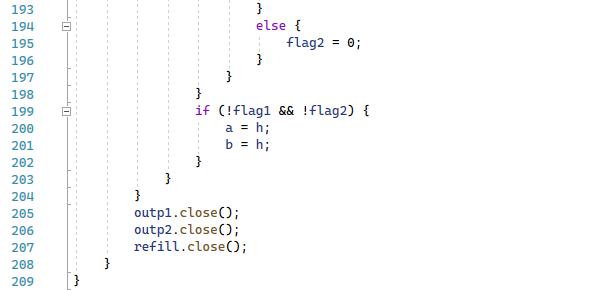
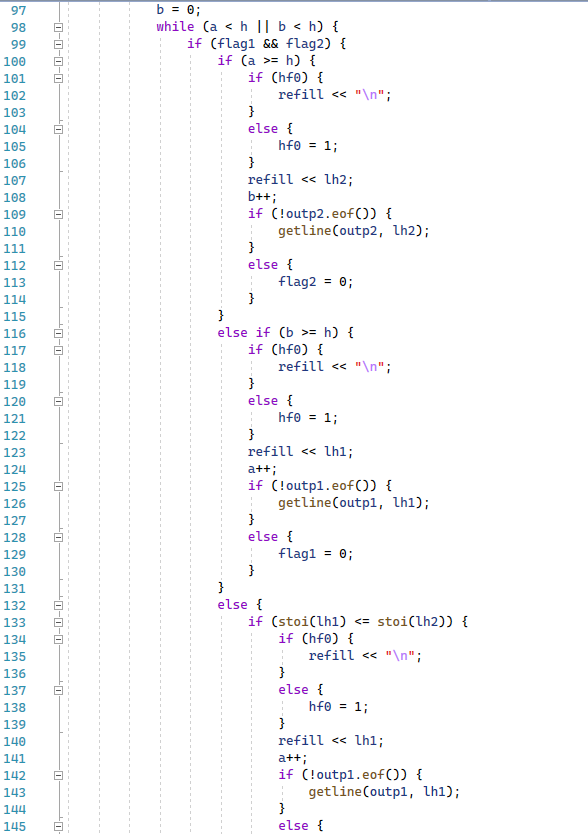
Header.h:



Source.cpp:

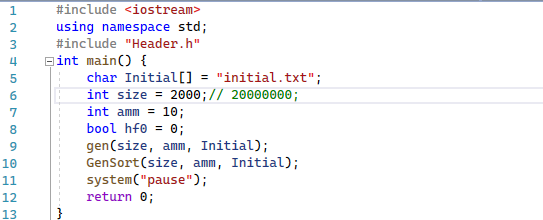




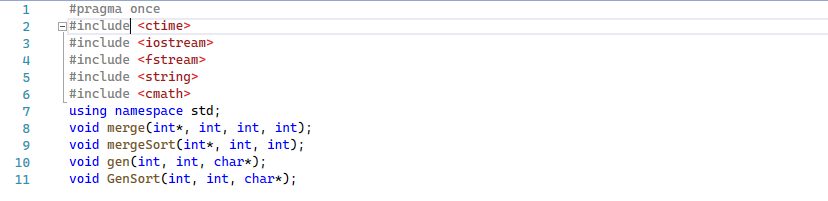


### Модифікований код

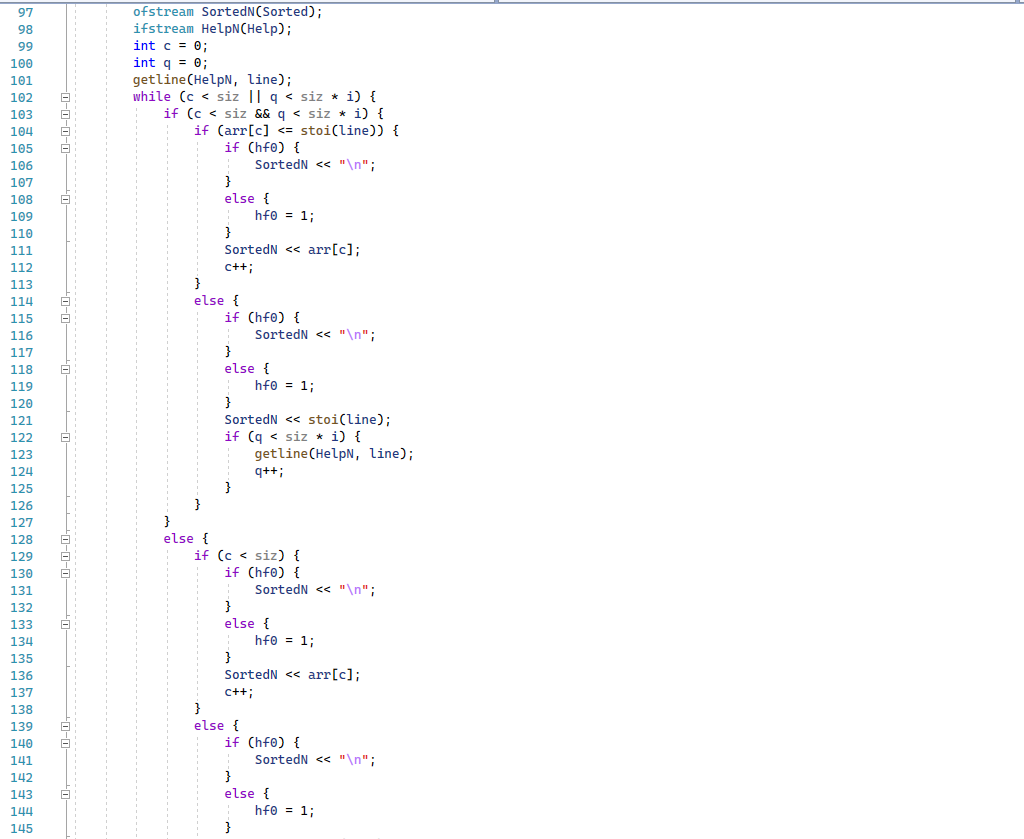
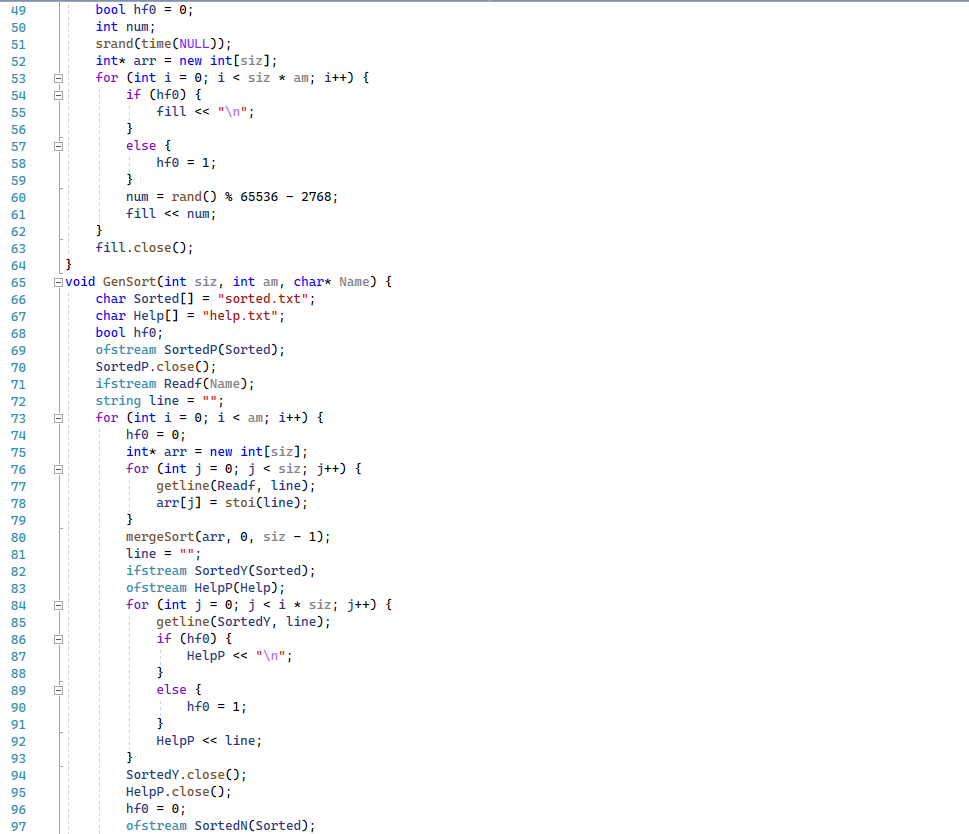
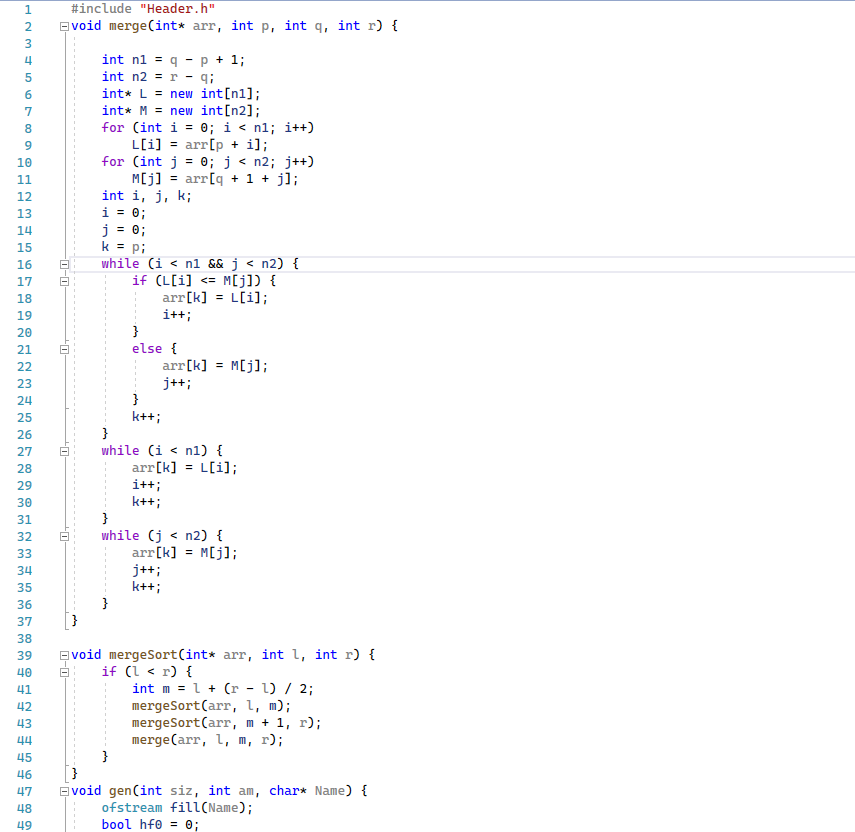
AP\_Laba\_1\_Mod.cpp:



Header.h:



Source.cpp:



Висновок

При виконанні даної лабораторної роботи ми створили та оптимізували алгоритм сортування прямим злиттям.

Модифікований алгоритм, на відміну від не модифікованого, розбиває файл на серії та сортує кожну серію окремо внутрішнім прямим злиттям, а потім об’єднує серію з відсортованою частиною файлу так, щоб елементи були впорядковані. Не модифікований алгоритм розбиває кожного разу файл на два і потім впорядковує маленькі групи елементів. Так, як виникає постійне зчитування й запис у файли, то комп’ютер витрачає більше часу на сортування одного й того самого об’єму даних, ніж модифікований, адже серія записується у файл лише після сортування серії.

Критерії оцінювання

У випадку здачі лабораторної роботи до 09.10.2022 включно максимальний бал дорівнює – 5. Після 09.10.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

* псевдокод алгоритму – 15%;
* програмна реалізація алгоритму – 40%;
* програмна реалізація модифікацій – 40%;
* висновок – 5%.