**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

направление специальности 1-40 05 01-12 Информационные системы

и технологии (в игровой индустрии)

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине: «Введение в разработку программного обеспечения»

на тему: «Моделирование и алгоритмизация как средства проектирования   
программного обеспечения»

Выполнил: студент гр. ИТИ-21

Королевич Н.Я.

Принял: преподаватель

Ястребов А.А.

Гомель 2024

**Цель работы**: изучить моделирование и алгоритмизацию как средства программного обеспечения.

**Задание:**

1. Описать блок-схему алгоритма пирамидальной сортировки.

2. На основе описанной блок-схемы реализовать алгоритм средствами языка Python.

3. Протестировать разработанный алгоритм при помощи модульных тестов.

4. Создать консольное приложение для демонстрации работы алгоритма.

5. Составить отчет о проделанной работе.

**Ход работы и результаты выполнения:**

Графическая схема алгоритма представлена на рисунке 1.

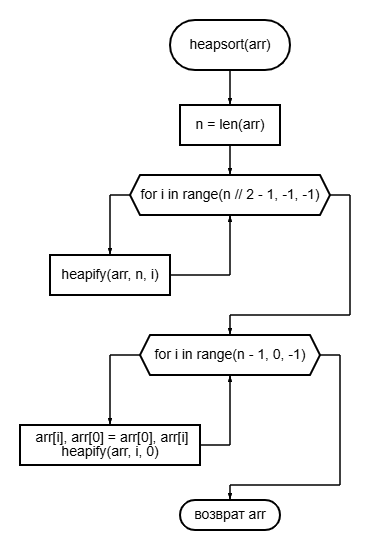


Рисунок 1 – Графическая схема алгоритма

Графическая схема алгоритма *heapify* представлена на рисунке 2.

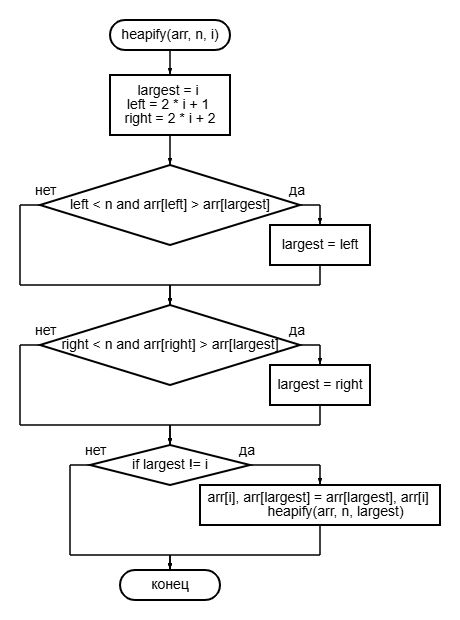


Рисунок 2 – Графическая схема алгоритма heapify

Результат выполнения задания 1 представлен на рисунке 3.

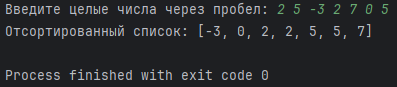


Рисунок 3 – Результат выполнения задания 1

Результат тестирования программы представлен на рисунке 4.

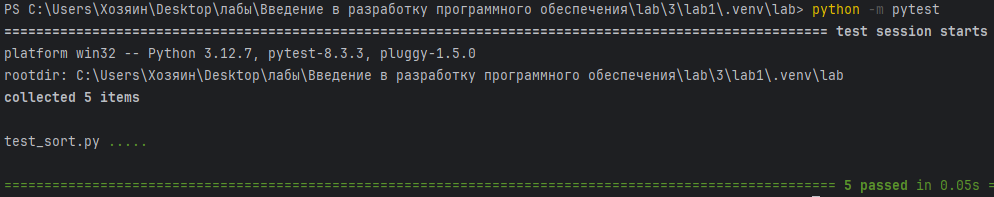


Рисунок 4 – Результат выполнения тестирования

Текст разработанной программы представлен в приложении А.

**Вывод:** в ходе лабораторной работы изучены моделирование и алгоритмизация как средства программного обеспечения. Описана блок-схема алгоритма пирамидальной сортировки, на основе описанной блок-схема и протестирован алгоритм средствами *Python*.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программ**

**«lab1.py»**

def input\_data() -> list[int]:  
 # Ввод данных от пользователя  
 data = list(map(int, input("Введите целые числа через пробел: ").split()))  
 return data  
  
def heapify(arr: list[int], n: int, i: int) -> None:  
 # Приведение подмассива к куче  
 largest = i  
 left = 2 \* i + 1  
 right = 2 \* i + 2  
  
 if left < n and arr[left] > arr[largest]:  
 largest = left  
  
 if right < n and arr[right] > arr[largest]:  
 largest = right  
  
 if largest != i:  
 arr[i], arr[largest] = arr[largest], arr[i]  
 heapify(arr, n, largest)  
  
def heapsort(arr: list[int]) -> list[int]:  
 n = len(arr)  
  
 # Построение кучи  
 for i in range(n // 2 - 1, -1, -1):  
 heapify(arr, n, i)  
  
 # Сортировка  
 for i in range(n - 1, 0, -1):  
 arr[i], arr[0] = arr[0], arr[i] # Перемещаем корень (максимум) в конец  
 heapify(arr, i, 0) # Восстанавливаем кучу  
  
 return arr  
  
def output\_result(result: list[int]) -> None:  
 # Вывод отсортированных данных на консоль  
 print("Отсортированный список:", result)  
  
# Основная функция для демонстрации работы алгоритма  
def main() -> None:  
 data = input\_data()  
 sorted\_data = heapsort(data)  
 output\_result(sorted\_data)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**«test\_sort.py»**

import pytest  
from lab1 import heapsort  
  
def test\_heapsort():  
 test\_data = [3, 1, 4, 1, 5, 9, 2, 6, 5]  
 expected\_result = sorted(test\_data)  
 assert heapsort(test\_data) == expected\_result  
  
def test\_empty\_list():  
 assert heapsort([]) == []  
  
def test\_single\_element():  
 assert heapsort([42]) == [42]  
  
def test\_sorted\_list():  
 assert heapsort([1, 2, 3, 4, 5]) == [1, 2, 3, 4, 5]  
  
def test\_reverse\_sorted\_list():  
 assert heapsort([5, 4, 3, 2, 1]) == [1, 2, 3, 4, 5]