

МИНОБРНАУКИ РОССИИ

**ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)**

**ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО**

**ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕСИТЕТ»**

**В Г.ТАГАНРОГЕ РОСТОВСКОЙ ОБЛАСТИ**

**ПИ (филиал) ДГТУ в г. Таганроге**

Факультет \_\_\_\_ \_\_Высшего образования\_\_ \_\_\_\_\_

(наименование факультета)

Кафедра \_ \_Технический сервис и информационные технологии \_

(наименование кафедры)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| И.о. зав. кафедрой | | | «\_ТСиИТ\_» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | \_\_И.А.Дмитриева\_ | |
| (подпись) | | | (И.О.Ф.) |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_г. | | | |

**ОТЧЕТ**

по \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Лабораторной работе № 1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

вид работы

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Константин Витальевич Матыцин

подпись, датаИ.О.Ф.

Обозначение отчета 520000 Группа ИВТ-2121

Направление \_\_09.03.01\_\_ \_\_\_ Информатика и вычислительная техника\_

коднаименование направления подготовки

Профиль \_\_\_\_\_Информатика и вычислительная техника\_\_\_\_\_\_\_\_\_

Преподаватель от кафедры \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность подпись, дата имя, отчество, фамилия

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата подпись преподавателя

Таганрог

2022

СОДЕРЖАНИЕ

|  |
| --- |
| 1. Введение………………..…...………………………………...…..…………3 |
| 1. Сортировка обменом…..….………………………………………..…….....4 |
| 1. Сортировка выбором…..………………………………...……………..…...6 |
| 1. Сортировка вставкой…..……………………………….…………………...8 |
| 1. Заключение……………...………………...………….…………………..…10 |
| 1. Список используемой литературы….…...………….…………………..…11 |

**Введение**

Пузырьковая сортировка — это алгоритм сортировки, который сравнивает два соседних элемента и меняет их местами до тех пор, пока они не окажутся в заданном порядке. Подобно движению пузырьков воздуха в воде, которые поднимаются на поверхность, каждый элемент массива движется к концу в каждой итерации. Поэтому его называют пузырьковой сортировкой.

Сортировка выбором — это алгоритм сортировки, который выбирает наименьший элемент из несортированного списка на каждой итерации и помещает этот элемент в начало несортированного списка.

Сортировка вставками — это алгоритм сортировки, который помещает несортированный элемент в подходящее место на каждой итерации. Сортировка вставками работает так же, как мы сортируем карты в руке в карточной игре. Мы предполагаем, что первая карта уже отсортирована, тогда мы выбираем несортированную карту. Если неотсортированная карта больше, чем карта в руке, она кладется справа, в противном случае — слева. Таким же образом берутся и другие несортированные карты и кладутся на свои места.

Аналогичный подход используется при сортировке вставками.

**Сортировка обменом**

Суть алгоритма состоит в следующем: сравниваются пары рядом стоящих элементов массива, если первый элемент пары меньше второго, то элементы меняются местами. После первого просмотра массива самый большой элемент встает на свое место, а маленькие по значению элементы на один шаг продвигаются к началу массива.

Ход работы:

Используя язык C напишем код, который будет сортировать группы цифр: {9, 5, 7, 3, 6, 4, 2}, {-20, 7, -16, 5, 10, -2, 3}

#include **<stdio.h>***// объявляем пузырьки***void** bubbleSort(**int** array[], **int** size) {  
  
 *// цикл бегающий по всем элементам массива* **for** (**int** step = 0; step < size - 1; ++step) {  
  
 *// цикл сравнивающий элементы массива* **for** (**int** i = 0; i < size - step - 1; ++i) {  
   
 *// сравниваем двух соседей* **if** (array[i] > array[i + 1]) {  
  
 *// замена при соблюдении условий* **int** temp = array[i];  
 array[i] = array[i + 1];  
 array[i + 1] = temp;  
 }  
 }  
 }  
}  
  
*// выводим массивчик***void** printArray(**int** array[], **int** size) {  
 **for** (**int** i = 0; i < size; ++i) {  
 printf(**"%d "**, array[i]);  
 }  
 printf(**"\n"**);  
}  
*// Исполнение***int** main() {  
 **int** data[] = {-20, 7, -16, 5, 10, -2, 3};  
 **int** data1[] = {9, 5, 7, 3, 6, 4, 2};  
 *// интересуемся у массива, насколько он длинный?* **int** size = **sizeof**(data) / **sizeof**(data[0]);  
 **int** size1 = **sizeof**(data1) / **sizeof**(data1[0]);  
 bubbleSort(data, size);  
 printf(**"Sorted array 1 in Acsending Order:\n"**);  
 printArray(data, size);  
 bubbleSort(data1, size1);  
 printf(**"Sorted array 2 in Acsending Order:\n"**);  
 printArray(data1, size1);  
}

Sorted array 1 in Acsending Order:

-20 -16 -2 3 5 7 10

Sorted array 2 in Acsending Order:

2 3 4 5 6 7 9

Версия программы для массивов случайных чисел

#include **<stdio.h>**#define SIZE (Указываем число от 100 до 10000)  
**int** d = 0;  
**int** m = 0;  
*// объявляем пузырьки***void** bubbleSort(**int** array[SIZE], **int** size) {  
  
 *// цикл бегающий по всем элементам массива* **for** (**int** step = 0; step < SIZE - 1; ++step) {  
  
 *// цикл сравнивающий элементы массива* **for** (**int** i = 0; i < SIZE - step - 1; ++i) {  
  
 m = m + 1;  
 *// сравниваем двух соседей* **if** (array[i] > array[i + 1]) {  
  
 *// замена при соблюдении условий* d = d+1;  
 **int** temp = array[i];  
 array[i] = array[i + 1];  
 array[i + 1] = temp;  
 }  
 }  
 }  
}  
  
*// выводим массивчик***void** printArray(**int** array[SIZE], **int** size) {  
 **for** (**int** i = 0; i < SIZE; ++i) {  
 printf(**"%d "**, array[i]);  
 }  
 printf(**"\n"**);  
}  
  
*// Исполнение***int** main() {  
 **int** data[SIZE];  
 *// интересуемся у массива, насколько он длинный?* **int** size = SIZE;  
 bubbleSort(data, size);  
 printf(**"Output:\n"**);  
 printArray(data, size);  
 printf(**"%d"**, m+d);  
  
}

**Сортировка выбором**

Сортировка выбором – возможно, самый простой в реализации алгоритм сортировки. Как и в большинстве других подобных алгоритмов, в его основе лежит операция сравнения. Сравнивая каждый элемент с каждым, и в случае необходимости производя обмен, метод приводит последовательность к необходимому упорядоченному виду.

Ход работы:

Используя язык C напишем код, который будет сортировать группы цифр: {9, 5, 7, 3, 6, 4, 2}, {-20, 7, -16, 5, 10, -2, 3}

#include **<stdio.h>***// функция для замены позиций элементов***void** swap(**int** \*a, **int** \*b) {  
 **int** temp = \*a;  
 \*a = \*b;  
 \*b = temp;  
}  
  
**void** selectionSort(**int** array[], **int** size) {  
 **for** (**int** step = 0; step < size - 1; step++) {  
 **int** min\_idx = step;  
 **for** (**int** i = step + 1; i < size; i++) {  
 *// Ищем минимальный элемент в каждом обороте цикла* **if** (array[i] < array[min\_idx])  
 min\_idx = i;  
 }  
 swap(&array[min\_idx], &array[step]);  
 }  
}  
  
*// выводим массивчик***void** printArray(**int** array[], **int** size) {  
 **for** (**int** i = 0; i < size; ++i) {  
 printf(**"%d "**, array[i]);  
 }  
 printf(**"\n"**);  
}  
  
*// Исполнение***int** main() {  
 **int** data[] = {-20, 7, -16, 5, 10, -2, 3};  
 **int** data1[] = {9, 5, 7, 3, 6, 4, 2};  
 *// интересуемся у массива, насколько он длинный?* **int** size = **sizeof**(data) / **sizeof**(data[0]);  
 **int** size1 = **sizeof**(data1) / **sizeof**(data1[0]);  
  
 selectionSort(data, size);  
 printf(**"Sorted array 1 in Acsending Order:\n"**);  
 printArray(data, size);  
 selectionSort(data1, size1);  
 printf(**"Sorted array 2 in Acsending Order:\n"**);  
 printArray(data1, size1);  
}

Sorted array 1 in Acsending Order:

-20 -16 -2 3 5 7 10

Sorted array 2 in Acsending Order:

2 3 4 5 6 7 9

Версия программы для массивов случайных чисел

#include **<stdio.h>**#define SIZE (Указываем число от 100 до 10000)  
**int** m = 0;  
**int** d = 0;  
  
*// функция для замены позиций элементов***void** swap(**int** \*a, **int** \*b) {  
 **int** temp = \*a;  
 \*a = \*b;  
 \*b = temp;  
}  
  
**void** selectionSort(**int** array[], **int** size) {  
 **for** (**int** step = 0; step < size - 1; step++) {  
 **int** min\_idx = step;  
 **for** (**int** i = step + 1; i < size; i++) {  
 *// Ищем минимальный элемент в каждом обороте цикла* m++;  
 **if** (array[i] < array[min\_idx])  
 min\_idx = i;  
 d++;  
 }  
 swap(&array[min\_idx], &array[step]);  
 }  
}  
  
*// выводим массивчик***void** printArray(**int** array[], **int** size) {  
 **for** (**int** i = 0; i < size; ++i) {  
 printf(**"%d "**, array[i]);  
 }  
 printf(**"\n"**);  
}  
  
*// Исполнение***int** main() {  
 **int** data[SIZE];  
 *// интересуемся у массива, насколько он длинный?* **int** size = **sizeof**(data) / **sizeof**(data[0]);  
 selectionSort(data, size);  
 printf(**"Output:\n"**);  
 printArray(data, size);  
 printf(**"%d \n"**,m+d);  
}

**Сортировка вставкой**

Сортировка вставками – простой алгоритм сортировки, преимущественно использующийся в учебном программировании. К положительной стороне метода относится простота реализации, а также его эффективность на частично упорядоченных последовательностях, и/или состоящих из небольшого числа элементов. Тем не менее, высокая вычислительная сложность не позволяет рекомендовать алгоритм в повсеместном использовании.

Ход работы:

Используя язык C напишем код, который будет сортировать группы цифр: {9, 5, 7, 3, 6, 4, 2}, {-20, 7, -16, 5, 10, -2, 3}

#include **<stdio.h>***// Печатаем массивчик***void** printArray(**int** array[], **int** size) {  
 **for** (**int** i = 0; i < size; i++) {  
 printf(**"%d "**, array[i]);  
 }  
 printf(**"\n"**);  
}  
**void** insertionSort(**int** array[], **int** size) {  
 **for** (**int** step = 1; step < size; step++) {  
 **int** key = array[step];  
 **int** j = step - 1;  
  
 *// Сравниваем каждый элемент со всеми элементами слева, перемещаем пока не добьемся условия* **while** (key < array[j] && j >= 0) {  
 array[j + 1] = array[j];  
 --j;  
 }  
 array[j + 1] = key;  
 }  
}  
*// Исполнение***int** main() {  
 **int** data[] = {-20, 7, -16, 5, 10, -2, 3};  
 **int** data1[] = {9, 5, 7, 3, 6, 4, 2};  
  
 *// интересуемся у массива, насколько он длинный?* **int** size = **sizeof**(data) / **sizeof**(data[0]);  
 **int** size1 = **sizeof**(data1) / **sizeof**(data1[0]);  
  
 insertionSort(data, size);  
 printf(**"Sorted array 1 in Acsending Order:\n"**);  
 printArray(data, size);  
 insertionSort(data1, size1);  
 printf(**"Sorted array 2 in Acsending Order:\n"**);  
 printArray(data1, size1);  
}

Sorted array 1 in Acsending Order:

-20 -16 -2 3 5 7 10

Sorted array 2 in Acsending Order:

2 3 4 5 6 7 9

Версия программы для массивов случайных чисел

#include **<stdio.h>**#define SIZE (Указываем число от 100 до 10000)  
**int** d = 0;  
**int** m = 0;  
*// Печатаем массивчик***void** printArray(**int** array[], **int** size) {  
 **for** (**int** i = 0; i < size; i++) {  
 printf(**"%d "**, array[i]);  
 }  
 printf(**"\n"**);  
}  
  
**void** insertionSort(**int** array[], **int** size) {  
 **for** (**int** step = 1; step < size; step++) {  
 **int** key = array[step];  
 **int** j = step - 1;  
 m++;  
 *// Сравниваем каждый элемент со всеми элементами слева, перемещаем пока не добьемся условия* **while** (key < array[j] && j >= 0) {  
 array[j + 1] = array[j];  
 d++;  
 --j;  
 }  
 array[j + 1] = key;  
 }  
}  
  
*// Исполнение***int** main() {  
 **int** data[SIZE];  
 **int** size = **sizeof**(data) / **sizeof**(data[0]);  
 *// интересуемся у массива, насколько он длинный?* insertionSort(data, size);  
 printf(**"Output:\n"**);  
 printArray(data, size);  
 printf(**"%d \n"**,m+d);  
  
}

**Сортировка массивов случайных чисел и вычисление зависимости суммы количества сравнений и перестановок от объема массива**

В ходе работы мы дали нашим программам массивы случайных чисел объемом 100, 200, 500, 1000, 5000 и 10000 чисел. Вот таблица замеров:

|  |  |  |  |
| --- | --- | --- | --- |
| Объем | Selection | Bubble | Insertion |
| *100* | 5 214 | 6 452 | 1 947 |
| *200* | 20 569 | 22 352 | 9 396 |
| *500* | 126 547 | 179 721 | 49 704 |
| *1000* | 503 379 | 724 134 | 209 575 |
| *5000* | 12 508 935 | 17 402 224 | 1 852 220 |
| *10000* | 50 006 270 | 58 448 652 | 2 905 997 |
| *Среднее знач.* | 10 528 485,67 | 12 797 255,83 | 838 139,8333 |

На основе этих данных построим график зависимости:

Вывод: В ходе работы мы научились сортировать массивы чисел тремя методами, вычислять количество сравнений и перестановок, а также строить графики зависимости сложности алгоритма. Эти данные позволили нам выяснить, что на маленьких массивах разница пренебрежительна, но когда объем данных исчисляется тысячами, очевидно самым быстрым методом является метод вставки, а самым медленным – пузырьковый

**Список используемой литературы**

1. Вирт, Н. Алгоритмы и структуры данных. Новая версия для Оберо- на [Электронный ресурс] : учебное пособие / Н. Вирт. — Электрон. дан.

— Москва : ДМК Пресс, 2010. — 272 с. — Режим доступа: https://e.lanbook.com/book/1261. — Загл. с экрана.

2. Мещеряков, Р.В. Методы программирования [Электронный ресурс]

: учебно-методическое пособие / Р.В. Мещеряков. — Электрон. дан. — Москва : ТУСУР, 2007. — 237 с. — Режим доступа: https://e.lanbook.com/book/11631. — Загл. с экрана.

3. Ален Голуб. ВЕРЕВКА ДОСТАТОЧНОЙ ДЛИНЫ, ЧТОБЫ… ВЫСТРЕЛИТЬ СЕБЕ В НОГУ Правила программирования на Си и Си++. Москва, 2021г.

4. Пермякова Наталья ВикторовнаИнформатика и программирование Часть I: Методические указания к лабораторным работам, практическим занятиям и организации самостоятельной работы для студентов направления «Программная инженерия» (уровень бакалавриата) / Н.В. Пермякова. — Томск, 2018 — 65 с.