ГУАП

# КАФЕДРА №43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  | 19.10.2023 |  | С.А. Рогачев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 5 |
| АЛГОРИТМЫ СОРТИРОВКИ |
| по курсу: АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4232 |  | 19.10.2023 |  | М.В.Куриш |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

1. **Цель работы**

Целью работы является изучение алгоритмов внутренней сортировки и получение практических навыков их использования, и анализа их сложности.

1. **Задание на лабораторную работу**

Использовать неупорядоченный массив A, содержащий n целочисленных элементов. Величина n определяется по согласованию с преподавателем. Дополнительно в программе должны быть реализованы следующие функции:

1. Поиск элемента либо по его порядковой позиции, либо по его содержимому;
2. Добавление/удаление элемента с последующей пересортировкой последовательности;
3. В программе должен быть реализован подсчет количества сравнений и перестановок, при осуществлении сортировки.

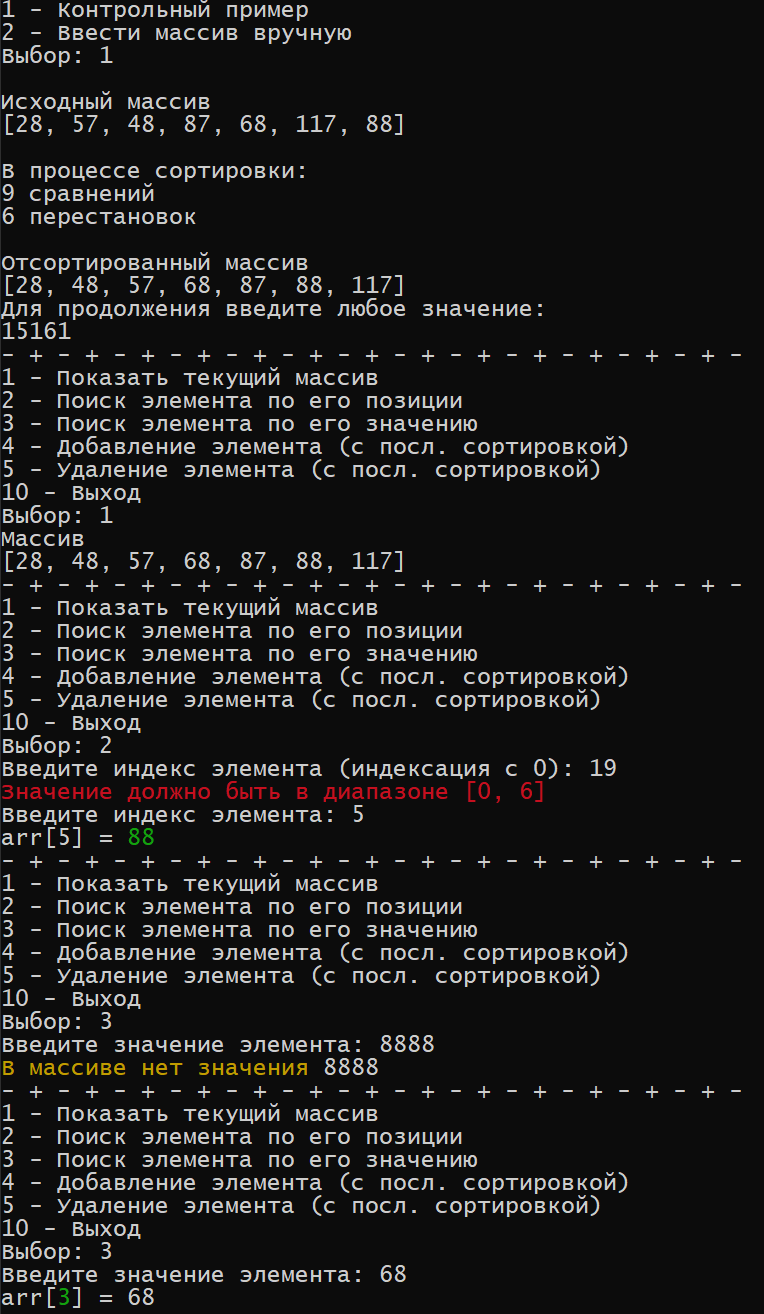
Вариант 22.

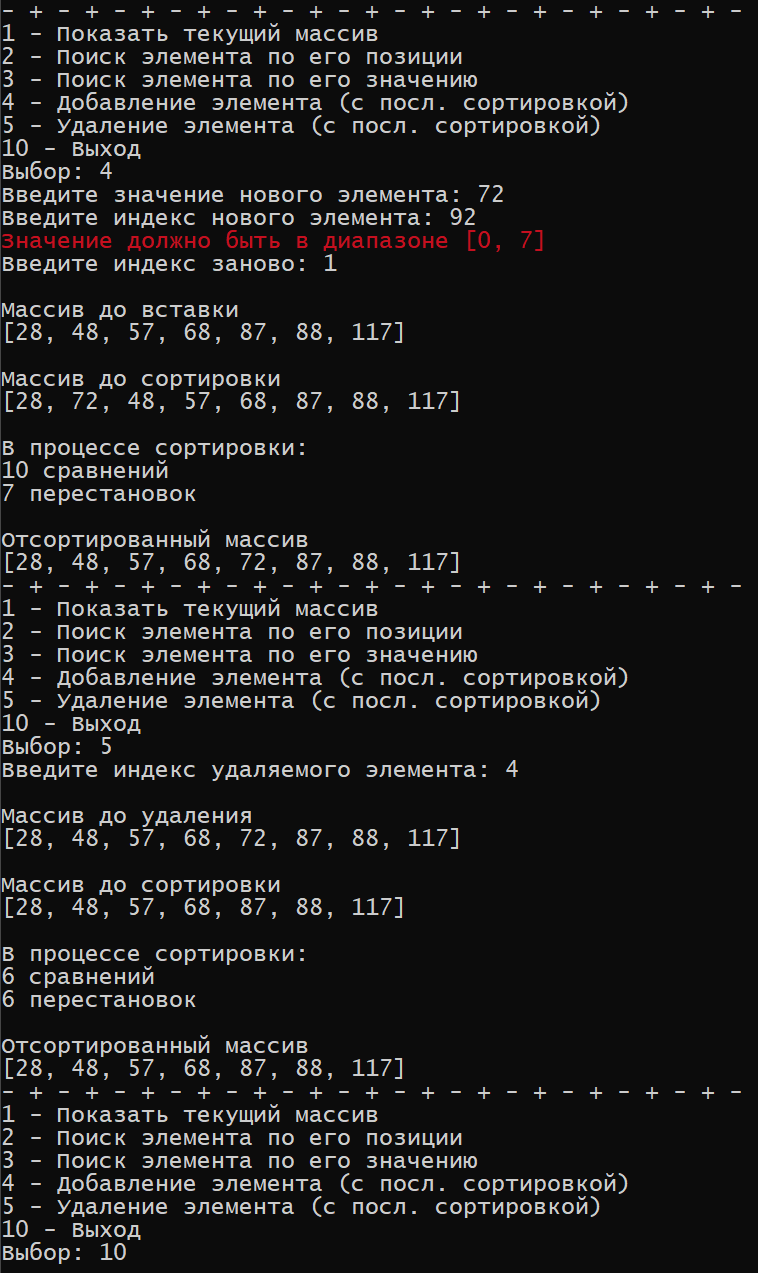
|  |  |  |
| --- | --- | --- |
| **№ вар.** | **Задание** | **Алгоритм сортировки** |
| 22 | Найти k-ое по порядку число среди элементов массива | Простым включением |

1. **Листинг программы**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  int intInput(string msg = "Введите заново: ") {  int a;  cin >> a;  while (!cin) {  cout << "\x1B[31mНеверный ввод.\033[0m (Ожидается число)\n";  cin.clear();  cin.ignore(999, '\n');  cout << msg;  cin >> a;  }  return a;  }  void showArray(int\* arr, int size, string header) {  cout << header << endl;  cout << "[";  for (int i = 0; i < size; i++) {  cout << arr[i];  if (i != size - 1)  cout << ", ";  }  cout << "]\n";  }  string sortSimpleInclusion(int\* arr, int size) {  int comparisonCount = 0;  int swapCount = 0;  for (int i = 1; i < size; i++) {  int temp = arr[i];  int j = i - 1;  while (arr[j] > temp && j >= 0) {  comparisonCount++;  arr[j + 1] = arr[j];  swapCount++;  j--;  }  comparisonCount++;  arr[j + 1] = temp;  swapCount++;  }  return "\nВ процессе сортировки:\n" + to\_string(comparisonCount) + " сравнений\n" + to\_string(swapCount) + " перестановок\n";  }  int main()  {  setlocale(LC\_ALL, "RUS");  int\* arr;  int arrSize;  cout << "1 - Контрольный пример\n";  cout << "2 - Ввести массив вручную\n";  cout << "Выбор: ";  int menuChoise = intInput();  while (menuChoise != 2 && menuChoise != 1) {  cout << "\x1B[31mНеверный выбор пункта меню.\033[0m\nПовторите попытку: ";  menuChoise = intInput();  }  if (menuChoise == 1) {  arrSize = 7;  arr = new int[arrSize];  for (int i = 0; i < arrSize; i++) {  arr[i] = ((i + 3) \* 5 - 1) \* (i % 2 + 2);  }  }  else {  cout << "Введите размер массива: ";  arrSize = intInput();  arr = new int[arrSize];  for (int i = 0; i < arrSize; i++) {  cout << "Введите значение элемента #" << i + 1 << ": ";  arr[i] = intInput("Введите значение элемента #" + to\_string(i + 1) + ": ");  }  }  showArray(arr, arrSize, "\nИсходный массив");  cout << sortSimpleInclusion(arr, arrSize);  showArray(arr, arrSize, "\nОтсортированный массив");  cout << "Для продолжения введите любое значение: \n";  cin.clear();  cin.ignore(999, '\n');  int garbage;  cin >> garbage;  bool exit = false;  int index;  int val;  int\* temp;  bool deleted;  bool found;  bool inserted;  while (!exit) {  cout << "- + - + - + - + - + - + - + - + - + - + - + - + - + -";  cout << "\n1 - Показать текущий массив\n";  cout << "2 - Поиск элемента по его позиции\n";  cout << "3 - Поиск элемента по его значению\n";  cout << "4 - Добавление элемента (с посл. сортировкой)\n";  cout << "5 - Удаление элемента (с посл. сортировкой)\n";  cout << "10 - Выход\n";  cout << "Выбор: ";  switch (intInput())  {  case 1:  showArray(arr, arrSize, "Массив");  break;  case 2:  cout << "Введите индекс элемента (индексация с 0): ";  index = intInput("Введите индекс элемента: ");  while (index < 0 || index >= arrSize) {  cout << "\x1B[31mЗначение должно быть в диапазоне [" << 0 << ", " << arrSize - 1 << "]\033[0m\n";  cout << "Введите индекс элемента: ";  index = intInput("Введите индекс элемента: ");  }  cout << "arr[" << index << "] = \x1B[32m" << arr[index] << "\033[0m\n";  break;  case 3:  cout << "Введите значение элемента: ";  val = intInput("Введите значение элемента: ");  found = false;  for (int i = 0; i < arrSize; i++) {  if (arr[i] == val) {  cout << "arr[\x1B[32m" << i << "\033[0m] = " << val << endl;  found = true;  break;  }  }  if (!found)  cout << "\x1B[33mВ массиве нет значения\033[0m " << val << endl;  break;  case 4:  cout << "Введите значение нового элемента: ";  val = intInput("Введите значение нового элемента: ");  cout << "Введите индекс нового элемента: ";  index = intInput("Введите индекс нового элемента: ");  while (index < 0 || index > arrSize) {  cout << "\x1B[31mЗначение должно быть в диапазоне [" << 0 << ", " << arrSize << "]\033[0m\n";  cout << "Введите индекс заново: ";  index = intInput("Введите индекс элемента: ");  }  showArray(arr, arrSize, "\nМассив до вставки");  arrSize++;  temp = new int[arrSize];  inserted = false;  for (int i = 0; i < arrSize; i++) {  if (inserted)  temp[i] = arr[i - 1];  else {  if (i == index) {  temp[i] = val;  inserted = true;  }  else  temp[i] = arr[i];  }  }  delete[] arr;  arr = temp;  showArray(arr, arrSize, "\nМассив до сортировки");  cout << sortSimpleInclusion(arr, arrSize);  showArray(arr, arrSize, "\nОтсортированный массив");  break;  case 5:  cout << "Введите индекс удаляемого элемента: ";  index = intInput("Введите индекс удаляемого элемента: ");  while (index < 0 || index >= arrSize) {  cout << "\x1B[31mЗначение должно быть в диапазоне [" << 0 << ", " << arrSize - 1 << "]\033[0m\n";  cout << "Введите индекс заново: ";  index = intInput("Введите индекс элемента: ");  }  showArray(arr, arrSize, "\nМассив до удаления");  temp = new int[arrSize - 1];  deleted = false;  for (int i = 0; i < arrSize; i++) {  if (deleted)  temp[i - 1] = arr[i];  else {  if (i == index)  deleted = true;  else  temp[i] = arr[i];  }  }  arrSize--;  delete[] arr;  arr = temp;  showArray(arr, arrSize, "\nМассив до сортировки");  cout << sortSimpleInclusion(arr, arrSize);  showArray(arr, arrSize, "\nОтсортированный массив");  break;  case 10:  exit = true;  break;  default:  cout << "\x1B[31mНеверно выбран пункт меню!\033[0m\n";  break;  }  }  } |

1. **Контрольный пример**

****

****

1. **Количество выполненных операций сравнения и перестановок элементов массива**

Для контрольного примера, при исходном массиве 

количество операций сравнения: ,

количество операций перестановок: .

1. **Временная и пространственная сложность алгоритма**

Алгоритм использует:

1. Пять переменных типа int

(Данные, передаваемые функции сортировки в качестве аргументов, как правило, не включаются в расчет пространственной сложности алгоритма).

Следовательно, пространственная сложность алгоритма определяется следующим образом:



Где  - константа, характеризующая объем памяти, отводимый под переменную целочисленного типа int.

Теоретическая пространственная сложность алгоритма составляет:



Где  - размер входного массива.

Теоретическую временную сложность алгоритма определим на основе анализа текста программы, реализующей алгоритм. Для начала рассчитаем теоретическую временную сложность алгоритма сортировки в худшем случае:



Где  - операции , имеющие временную сложность “1”.

В лучшем же случае теоретическая временная сложность:



1. **Выводы**

Изучены алгоритмы внутренней сортировки и получены практические навыки их использования, и анализа их сложности. Реализована сортировка простым включением.