**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра АПУ**

**ОТЧЕТ**

**по лабораторной работе №2**

**на тему: «Улучшенные методы сортировки»**

**по дисциплине «Алгоритмы и структуры данных»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3391 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Николаев В.Ю. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Ряскова Е.Б. |

Санкт-Петербург

2024 г.

## Цель работы

Получить практические навыки работы с улучшенными методами сортировки массивов структур.

## Задачи

1. Добавить в программу, полученную в результате выполнения лабораторной работы №1 два улучшенных метода сортировки: быстрая сортировка, пирамидальная сортировка.
2. Подсчитать и вывести на экран количество сравнений и обменов для каждого добавленного метода сортировки.
3. Обеспечить корректность ввода данных пользователем через меню программы.

## Описание программы

Программа реализует структуру данных «Студент», включающую следующие поля:

* ФИО
* Курс
* Факультет

## Алгоритмы сортировки

1. Сортировка обменом:

|  |
| --- |
| void bubble\_sort(const Student\* students, int student\_count) {  Student\* temp\_students = new Student[student\_count];  for (int i = 0; i < student\_count; ++i) {  temp\_students[i] = students[i];  }   int compares = 0, swaps = 0;  bool swapped;   for (int i = 0; i < student\_count - 1; i++) {  swapped = false;  for (int j = 0; j < student\_count - i - 1; j++) {  compares++;  if (temp\_students[j].course < temp\_students[j + 1].course) {  std::swap(temp\_students[j], temp\_students[j + 1]);  swaps++;  swapped = true;  }  }  if (!swapped) break;  }   print\_students(temp\_students, student\_count);  std::cout << "Пузырьковая сортировка: сравнений - " << compares << ", обменов - " << swaps << ".\n";   delete[] temp\_students; } |

1. Сортировка выбором:

|  |
| --- |
| void selection\_sort(const Student\* students, int student\_count) {  Student\* temp\_students = new Student[student\_count];  for (int i = 0; i < student\_count; ++i) {  temp\_students[i] = students[i];  }   int compares = 0, swaps = 0;  for (int i = 0; i < student\_count - 1; i++) {  int min\_idx = i;  for (int j = i + 1; j < student\_count; j++) {  compares++;  if (temp\_students[j].course > temp\_students[min\_idx].course) {  min\_idx = j;  }  }  if (min\_idx != i) {  std::swap(temp\_students[min\_idx], temp\_students[i]);  swaps++;  }  }   print\_students(temp\_students, student\_count);  std::cout << "Сортировка выбором: сравнений - " << compares << ", обменов - " << swaps << ".\n";   delete[] temp\_students; } |

1. Быстрая сортировка:

|  |
| --- |
| void quick\_sort\_helper(Student\* students, int low, int high, int& compares, int& swaps) {  if (low < high) {  Student pivot = students[high];  int i = (low - 1);  for (int j = low; j < high; j++) {  compares++;  if (students[j].course >= pivot.course) {  i++;  std::swap(students[i], students[j]);  swaps++;  }  }  std::swap(students[i + 1], students[high]);  swaps++;  int pi = i + 1;  quick\_sort\_helper(students, low, pi - 1, compares, swaps);  quick\_sort\_helper(students, pi + 1, high, compares, swaps);  }  }  void quick\_sort(const Student\* students, int student\_count) {  Student\* temp\_students = new Student[student\_count];  for (int i = 0; i < student\_count; ++i) {  temp\_students[i] = students[i];  }  int compares = 0, swaps = 0;  quick\_sort\_helper(temp\_students, 0, student\_count - 1, compares, swaps);  print\_students(temp\_students, student\_count);  std::cout << "Быстрая сортировка: сравнений - " << compares << ", обменов - " << swaps << ".\n";  delete[] temp\_students;  } |

1. Пирамидальная сортировка:

|  |
| --- |
| void heapify(Student\* students, int n, int i, int& compares, int& swaps) {  int smallest = i;  int left = 2 \* i + 1;  int right = 2 \* i + 2;  compares++;  if (left < n && students[left].course < students[smallest].course)  smallest = left;  compares++;  if (right < n && students[right].course < students[smallest].course)  smallest = right;  if (smallest != i) {  std::swap(students[i], students[smallest]);  swaps++;  heapify(students, n, smallest, compares, swaps);  }  }  void heap\_sort(const Student\* students, int student\_count) {  Student\* temp\_students = new Student[student\_count];  for (int i = 0; i < student\_count; ++i) {  temp\_students[i] = students[i];  }  int compares = 0, swaps = 0;  for (int i = student\_count / 2 - 1; i >= 0; i--)  heapify(temp\_students, student\_count, i, compares, swaps);  for (int i = student\_count - 1; i > 0; i--) {  std::swap(temp\_students[0], temp\_students[i]);  swaps++;  heapify(temp\_students, i, 0, compares, swaps);  }  print\_students(temp\_students, student\_count);  std::cout << "Пирамидальная сортировка: сравнений - " << compares << ", обменов - " << swaps << ".\n";  delete[] temp\_students;  } |

## Меню программы

Программа имеет следующее меню для взаимодействия с пользователем:

1. Ввод данных
2. Вывод исходных данных
3. Сортировка пузырьком
4. Сортировка выбором
5. Быстрая сортировка
6. Пирамидальная сортировка
7. Выход

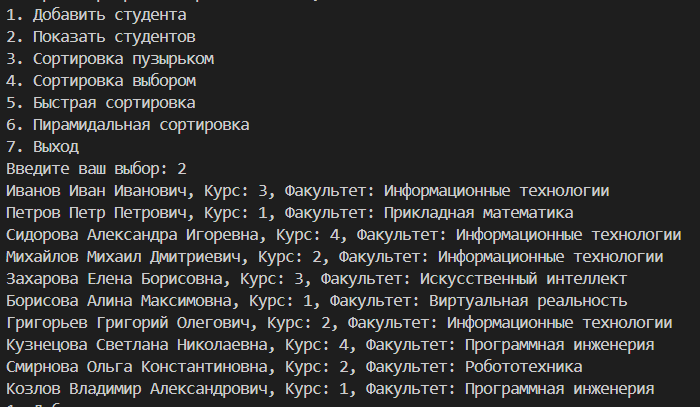
Кода меню:

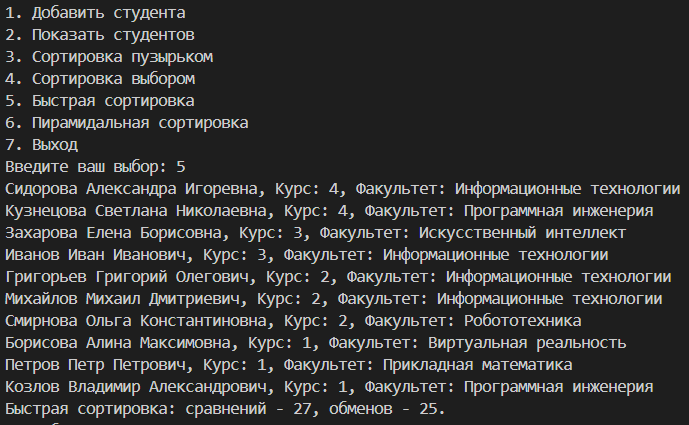
|  |
| --- |
| int main() {      int max\_students;      std::cout << "Введите максимальное количество студентов: ";      std::cin >> max\_students;      std::cin.ignore();      Student\* students = new Student[max\_students];      int student\_count = 0;      char command;      bool running = true;      while (running) {          std::cout << "1. Добавить студента\n"                       "2. Показать студентов\n"                       "3. Сортировка пузырьком\n"                       "4. Сортировка выбором\n"                       "5. Быстрая сортировка\n"                       "6. Пирамидальная сортировка\n"                       "7. Выход\n"                       "Введите ваш выбор: ";          std::cin >> command;          std::cin.ignore();          switch (command) {              case '1':                  add\_student(students, student\_count, max\_students);                  break;              case '2':                  print\_students(students, student\_count);                  break;              case '3':                  bubble\_sort(students, student\_count);                  break;              case '4':                  selection\_sort(students, student\_count);                  break;              case '5':                  quick\_sort(students, student\_count);                  break;              case '6':                  heap\_sort(students, student\_count);                  break;              case '7':                  running = false;                  break;              default:                  std::cout << "Неверный ввод, попробуйте снова.\n";          }      }      delete[] students; // Освобождение выделенной памяти      return 0;  } |

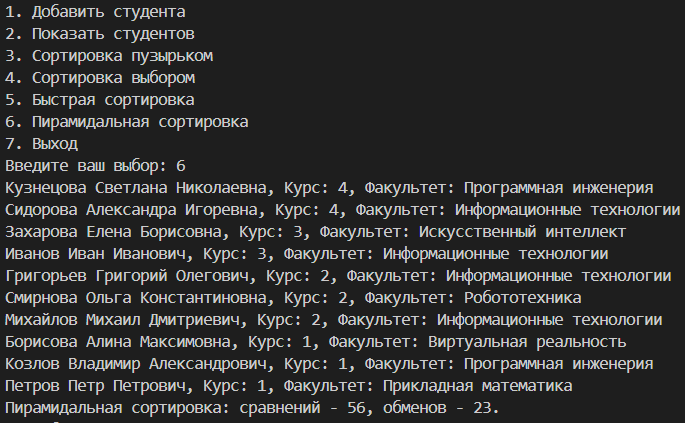
## Результаты работы программы

Пример ввода данных:  


|  |  |  |
| --- | --- | --- |
| ФИО | Курс | Факультет |
| Иванов Иван Иванович | 3 | Информационные технологии |
| Петров Петр Петрович | 1 | Прикладная математика |
| Сидорова Александра Игоревна | 4 | Информационные технологии |
| Михайлов Михаил Дмитриевич | 2 | Информационные технологии |
| Захарова Елена Борисовна | 3 | Искусственный интеллект |
| Борисова Алина Максимовна | 1 | Виртуальная реальность |
| Григорьев Григорий Олегович | 2 | Информационные технологии |
| Кузнецова Светлана Николаевна | 4 | Программная инженерия |
| Смирнова Ольга Константиновна | 2 | Робототехника |
| Козлов Владимир Александрович | 1 | Программная инженерия |

Исходный массив студентов: 

После быстрой сортировки: 

После пирамидальной выбором: 

## Выводы

В ходе выполнения лабораторной работы были разработаны два алгоритма сортировки для массива структур. Оба метода успешно отсортировали студентов по убыванию курса. Сортировка выбором показала меньшее количество сравнений, чем сортировка обменом.