1. void shellSort(A\* array, int size) - Объявляется шаблонная функция shellSort, которая принимает на вход указатель на массив типа A и его размер.

2. for (int step = size / 2; step > 0; step /= 2) - Начинается цикл, который инициализирует переменную шага step равную половине размера массива, и уменьшая ее вдвое на каждой итерации, пока step больше нуля. Этот подход позволяет устанавливать начальный интервал сравнения элементов.

3. for (int i = step; i < size; i++) - Внутренний цикл, начинающийся с шага и до конца массива. Мы рассматриваем элементы параллельно с шагом для сравнения.

4. tmp = array[i]; - Временно сохраняем текущий элемент массива.

5. for (j = i; j >= step; j -= step) - Внутренний цикл, который начинается с текущего элемента i и перемещается влево на шаг, сравнивая элементы на расстоянии шага друг от друга.

6. if (tmp < array[j - step]) - Если текущий элемент меньше элемента на шаг назад, то запускается блок действий для перестановки.

7. array[j] = array[j - step]; - Значение элемента с индексом j изменяется на значение элемента, находящегося на шаг влево.

8. else - В противном случае.

9. break; - Прерывается цикл, если условие внутри него больше не выполняется.

10. array[j] = tmp; - Текущему элементу присваивается сохраненное ранее значение tmp.

Этот алгоритм группирует элементы в массиве, чтобы уменьшить количество сдвигов элементов при сортировке, что улучшает производительность по сравнению с обычной сортировкой вставками.

ВТОРАЯ:

1. struct Weather { int day, temperature; }; - Определение структуры Weather, которая содержит два поля: день и температуру.

2. void shellSort(vector<Weather>& array, int size) - Объявление функции shellSort, принимающей ссылку на вектор структур Weather и его размер.

3. int temp1, temp2, j; - Объявление временных переменных для хранения значений температуры и дня.

4. for (int step = size / 2; step > 0; step /= 2) - Цикл определения шага для сортировки Шелла, начиная со среднего значения и уменьшая его вдвое на каждой итерации.

5. for (int i = step; i < size; i++) - Внешний цикл, который начинается с шага и до конца вектора, чтобы сравнивать элементы на определенном расстоянии.

6. temp1 = array[i].temperature; temp2 = array[i].day; - Сохранение текущих значений температуры и дня для текущего элемента.

7. for (j = i; j >= step; j -= step) - Внутренний цикл, который начинается с текущего индекса и движется влево на шаг, чтобы сравнивать элементы.

8. if (temp1 < array[j - step].temperature) - Если температура текущего элемента меньше температуры элемента на расстоянии шага, то запускается блок сортировки.

9. array[j].temperature = array[j - step].temperature; array[j].day = array[j - step].day; - Замена значений температуры и дня текущего элемента на значения элемента с шагом назад.

10. else - В противном случае.

11. break; - Прерывание цикла в случае, если условие не выполняется.

12. array[j].temperature = temp1; array[j].day = temp2; - Присваивание текущему элементу сохраненных ранее значений температуры и дня.

13. int main() - Основная функция программы.

14. setlocale(LC\_ALL, "Rus"); - Установка локали вывода.

15. Ввод размера вектора и его элементов с помощью цикла for с запросами о дне и температуре.

16. shellSort(weather, size); - Вызов функции сортировки Шелла для массива Weather.

17. Вывод отсортированных значений дня и температуры после сортировки.

Данный код производит сортировку массива структур Weather методом Шелла по убыванию температуры, сохраняя соответствие между днем и температурой.

ХОАР::

1. void quicksort(int\* items, int left, int right) - Объявляется функция quicksort, которая принимает указатель на массив items, левую и правую границы сортируемого участка.

2. int i, j, pivot, tmp; - Объявляются переменные для индексов i, j, опорного элемента pivot и временной переменной для обмена tmp.

3. i = left; - Устанавливается начальное значение левой границы.

4. j = right; - Устанавливается начальное значение правой границы.

5. pivot = items[(left + right) / 2]; - Выбирается опорный элемент, который находится примерно посередине выбранного участка массива.

6. do { - Начинается цикл выполнения.

7. while ((items[i] < pivot) && (i < right)) i++; - Увеличиваем индекс i, пока элементы слева от опорного меньше него.

8. while ((pivot < items[j]) && (j > left)) j--; - Уменьшаем индекс j, пока элементы справа от опорного больше него.

9. if (i <= j) { - Если индекс i меньше или равен j, выполняется обмен значениями.

10. tmp = items[i]; items[i] = items[j]; items[j] = tmp; i++; j--; - Элементы itemsi и itemsj меняются местами, затем индексы увеличиваются и уменьшаются соответственно.

11. } while (i <= j); - Цикл продолжается, пока i не превысит j.

12. if (left < j) quicksort(items, left, j); - Рекурсивный вызов quicksort для левой части массива (от начала до j).

13. if (i < right) quicksort(items, i, right); - Рекурсивный вызов quicksort для правой части массива (от i до конца).

Алгоритм быстрой сортировки разделяет массив на две части относительно опорного элемента, сортируя каждую из них отдельно. Рекурсивные вызовы функции quicksort выполняются до тех пор, пока все участки массива не окажутся отсортированными.

ВТОРАЯ:

1. stk.push(right); - Помещение правой границы массива в стек.

2. do { - Начало цикла выполнения.

3. right = stk.top(); - Получение правой границы массива из верхушки стека.

4. stk.pop(); - Удаление правой границы массива из стека.

5. left = stk.top(); - Получение левой границы массива из верхушки стека.

6. stk.pop(); - Удаление левой границы массива из стека.

7. i = left; j = right; - Установка начальных значений индексов.

8. pivot = items[(right + left) / 2]; - Выбор опорного элемента, как серединного между правой и левой границами.

9. Продолжаются операции сортировки, аналогичные предыдущему участку кода, с учетом новых границ и опорного элемента.

10. if (i <= j) { - Если индексы не пересекаются.

11. tmp = items[i]; items[i] = items[j]; items[j] = tmp; i++; j--; - Обмен элементами и увеличение левого и уменьшение правого индексов соответственно.

12. } while (i <= j); - Цикл продолжается, пока индексы не пересекутся.

13. if (left < j) { - Если левая граница меньше j.

14. stk.push(left); stk.push(j); - Помещение левой и j границ в стек для дальнейшей обработки.

15. if (i < right) { - Если i меньше правой границы.

16. stk.push(i); stk.push(right); - Помещение i и правой границы в стек.

17. } while (!stk.empty()); - Завершение цикла, когда стек пустой.

Этот код расширяет алгоритм быстрой сортировки так, чтобы он мог работать с кусками массива на основании границ, храня их в стеке для дальнейшей обработки. Код продолжает разделение и сортировку участков массива до завершения всей сортировки.

ТРЕТЬЯ:

1. Объявляются необходимые библиотеки, включая iostream, stack, vector и string.

2. Определяется структура Spisok, содержащая два поля - lang (язык программирования) и fam (фамилия).

3. Объявляется функция quicksort, принимающая на вход вектор структур Spisok items, левую left и правую right границы для сортировки.

4. Создается стек stk, и в него помещаются начальные границы left и right для обработки с использованием стека.

5. Начинается цикл, в котором правая граница извлекается из стека, затем левая граница, и инициализируются переменные i, j и опорный элемент pivot.

6. Во внутреннем цикле while осуществляется сравнение фамилий элементов массива для нахождения i и j, которые затем обменяются местами в случае необходимости. Дополнительно обмениваются также языки программирования.

7. Если левая граница меньше j, то левая и j граница добавляются в стек для дальнейшей сортировки.

8. Если i меньше правой границы, то i и правая граница добавляются в стек.

9. Выполняется цикл while до тех пор, пока стек не пустой, чтобы дополнительно обработать элементы из стека.

10. Настраивается локаль для вывода на русском языке и кодировка на Windows-1251.

11. Пользователю предлагается ввести количество элементов списка.

12. Создается вектор array размера n и заполняется информацией о языке программирования и фамилии для каждого элемента.

13. Вызывается функция quicksort для сортировки массива по фамилиям.

14. Выводится отсортированный массив на экран.

Этот код сортирует вектор структур Spisok по фамилии с использованием алгоритма быстрой сортировки. Он позволяет пользователю ввести данные о языке программирования и фамилии, сортирует их по фамилии и выводит результат на экран.