Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Тема: «Внешние сортировки в С++»

Выполнил

Студент РИС-24-1б

Конькова С. С.

Проверил

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2025

**Анализ**

Естественное

1. Нахождение последовательностей: Пройдите по массиву и найдите все уже отсортированные последовательности (группы элементов, которые уже находятся в порядке).
2. Слияние последовательностей: Слейте найденные последовательности в один отсортированный массив. Используйте метод слияния, как в сортировке слиянием.
3. Повторение: Повторяйте процесс, пока не останется одна отсортированная последовательность.
4. Завершение: Верните окончательно отсортированный массив.

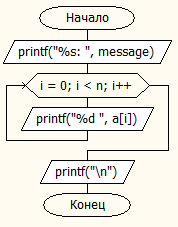
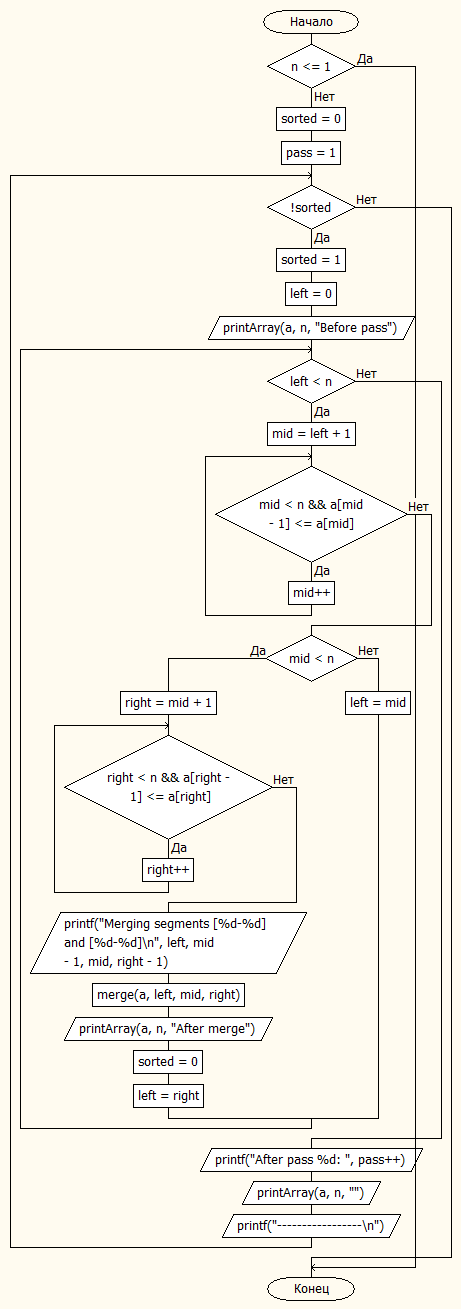
Сбалансированная сортировка

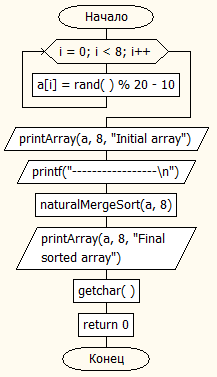
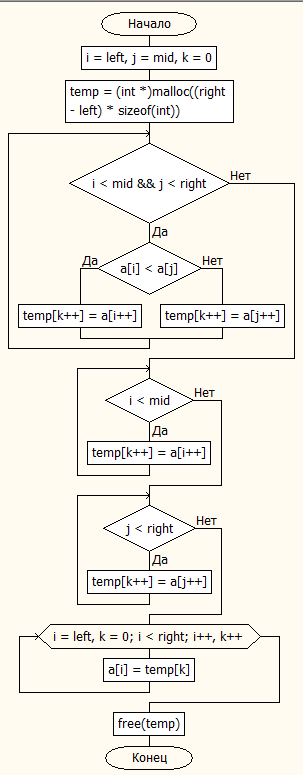
1. Нахождение последовательностей: Пройдите по массиву и найдите все уже отсортированные последовательности (группы элементов, которые уже находятся в порядке).
2. Слияние последовательностей: Слейте найденные последовательности в один отсортированный массив. Используйте метод слияния, как в сортировке слиянием.
3. Повторение: Повторяйте процесс, пока не останется одна отсортированная последовательность.
4. Завершение: Верните окончательно отсортированный массив.

Многофазная

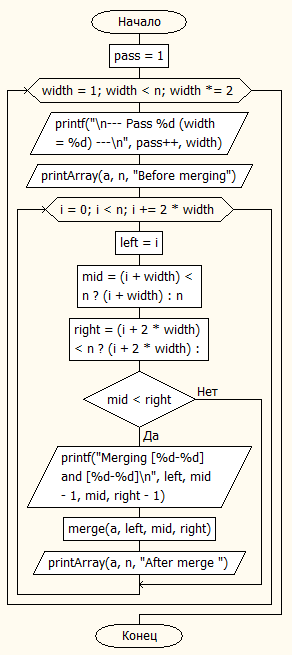
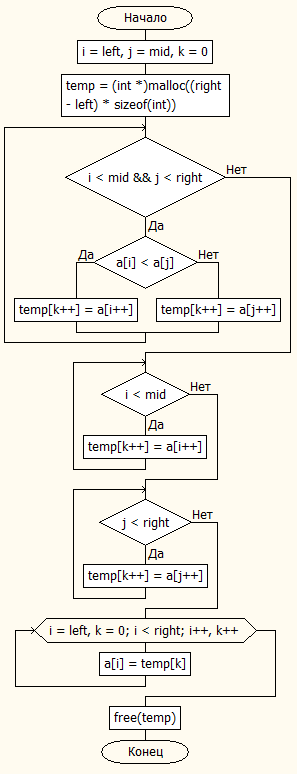
1. Нахождение последовательностей: Пройдите по массиву и найдите все уже отсортированные последовательности (группы элементов, которые уже находятся в порядке).
2. Слияние последовательностей: Слейте найденные последовательности в один отсортированный массив. Используйте метод слияния, как в сортировке слиянием.
3. Повторение: Повторяйте процесс, пока не останется одна отсортированная последовательность.
4. Завершение: Верните окончательно отсортированный массив.

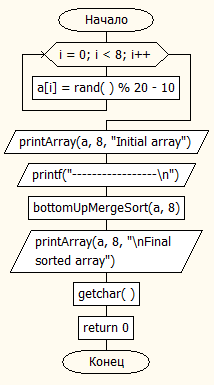
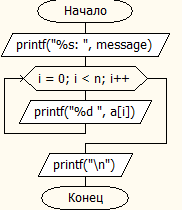
**Блок-схемы**

1. Естественное слияние

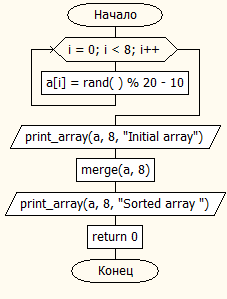
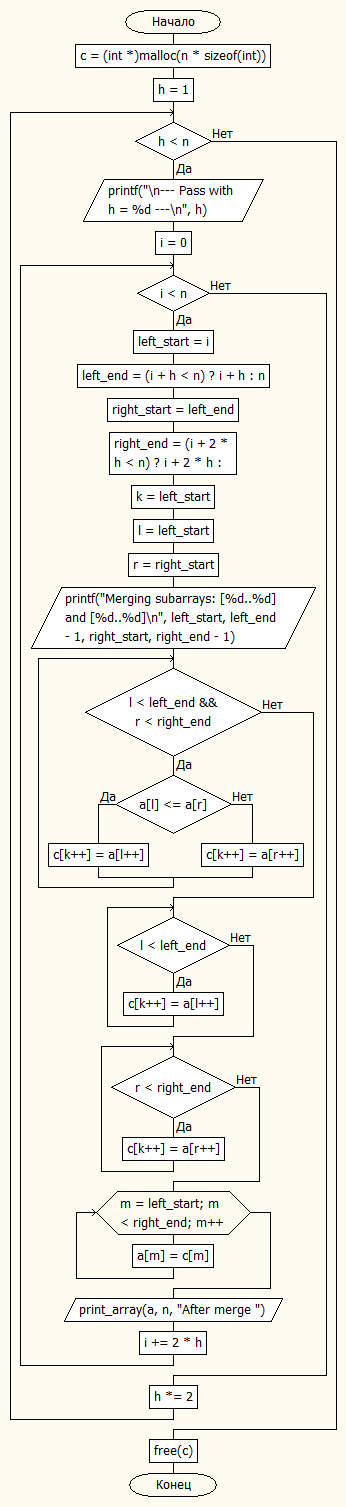
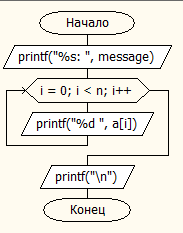


1. Сбалансированное слияние





1. Многофазная



**Коды программ**

1. Естественное слияние

#include <stdio.h>

#include <stdlib.h>

void printArray(int\* a, int n, const char\* message) {

printf("%s: ", message);

for (int i = 0; i < n; i++) {

printf("%d ", a[i]);

}

printf("\n");

}

void merge(int\* a, int left, int mid, int right) {

int i = left, j = mid, k = 0;

int\* temp = (int\*)malloc((right - left) \* sizeof(int));

while (i < mid && j < right) {

if (a[i] < a[j]) {

temp[k++] = a[i++];

} else {

temp[k++] = a[j++];

}

}

while (i < mid) {

temp[k++] = a[i++];

}

while (j < right) {

temp[k++] = a[j++];

}

for (i = left, k = 0; i < right; i++, k++) {

a[i] = temp[k];

}

free(temp);

}

void naturalMergeSort(int\* a, int n) {

if (n <= 1) return;

int left, mid, right;

int sorted = 0;

int pass = 1;

while (!sorted) {

sorted = 1;

left = 0;

printArray(a, n, "Before pass");

while (left < n) {

mid = left + 1;

while (mid < n && a[mid - 1] <= a[mid]) {

mid++;

}

if (mid < n) {

right = mid + 1;

while (right < n && a[right - 1] <= a[right]) {

right++;

}

printf("Merging segments [%d-%d] and [%d-%d]\n", left, mid-1, mid, right-1);

merge(a, left, mid, right);

printArray(a, n, "After merge");

sorted = 0;

left = right;

} else {

left = mid;

}

}

printf("After pass %d: ", pass++);

printArray(a, n, "");

printf("-----------------\n");

}

}

int main() {

int a[8];

for (int i = 0; i < 8; i++)

a[i] = rand() % 20 - 10;

printArray(a, 8, "Initial array");

printf("-----------------\n");

naturalMergeSort(a, 8);

printArray(a, 8, "Final sorted array");

getchar();

return 0;

}

1. Сбалансированное слияние

#include <stdio.h>

#include <stdlib.h>

void printArray(int\* a, int n, const char\* message) {

printf("%s: ", message);

for (int i = 0; i < n; i++) {

printf("%d ", a[i]);

}

printf("\n");

}

void merge(int\* a, int left, int mid, int right) {

int i = left, j = mid, k = 0;

int\* temp = (int\*)malloc((right - left) \* sizeof(int));

while (i < mid && j < right) {

if (a[i] < a[j]) {

temp[k++] = a[i++];

} else {

temp[k++] = a[j++];

}

}

while (i < mid) {

temp[k++] = a[i++];

}

while (j < right) {

temp[k++] = a[j++];

}

for (i = left, k = 0; i < right; i++, k++) {

a[i] = temp[k];

}

free(temp);

}

void bottomUpMergeSort(int\* a, int n) {

int width, i;

int pass = 1;

for (width = 1; width < n; width \*= 2) {

printf("\n--- Pass %d (width = %d) ---\n", pass++, width);

printArray(a, n, "Before merging");

for (i = 0; i < n; i += 2 \* width) {

int left = i;

int mid = (i + width) < n ? (i + width) : n;

int right = (i + 2 \* width) < n ? (i + 2 \* width) : n;

if (mid < right) { // Если есть что сливать

printf("Merging [%d-%d] and [%d-%d]\n", left, mid-1, mid, right-1);

merge(a, left, mid, right);

printArray(a, n, "After merge ");

}

}

}

}

int main() {

int a[8];

for (int i = 0; i < 8; i++)

a[i] = rand() % 20 - 10;

printArray(a, 8, "Initial array");

printf("-----------------\n");

bottomUpMergeSort(a, 8);

printArray(a, 8, "\nFinal sorted array");

getchar();

return 0;

}

1. Многофазная

#include <stdio.h>

#include <stdlib.h>

void print\_array(int\* a, int n, const char\* message) {

printf("%s: ", message);

for (int i = 0; i < n; i++) {

printf("%d ", a[i]);

}

printf("\n");

}

void merge(int\* a, int n) {

int\* c = (int\*)malloc(n \* sizeof(int));

int h = 1;

while (h < n) {

printf("\n--- Pass with h = %d ---\n", h);

int i = 0;

while (i < n) {

int left\_start = i;

int left\_end = (i + h < n) ? i + h : n;

int right\_start = left\_end;

int right\_end = (i + 2\*h < n) ? i + 2\*h : n;

int k = left\_start;

int l = left\_start;

int r = right\_start;

printf("Merging subarrays: [%d..%d] and [%d..%d]\n",

left\_start, left\_end-1, right\_start, right\_end-1);

while (l < left\_end && r < right\_end) {

if (a[l] <= a[r]) {

c[k++] = a[l++];

} else {

c[k++] = a[r++];

}

}

while (l < left\_end) {

c[k++] = a[l++];

}

while (r < right\_end) {

c[k++] = a[r++];

}

// Copy merged part back to original array

for (int m = left\_start; m < right\_end; m++) {

a[m] = c[m];

}

print\_array(a, n, "After merge ");

i += 2 \* h;

}

h \*= 2;

}

free(c);

}

int main() {

int a[8];

for (int i = 0; i < 8; i++)

a[i] = rand() % 20 - 10;

print\_array(a, 8, "Initial array");

merge(a, 8);

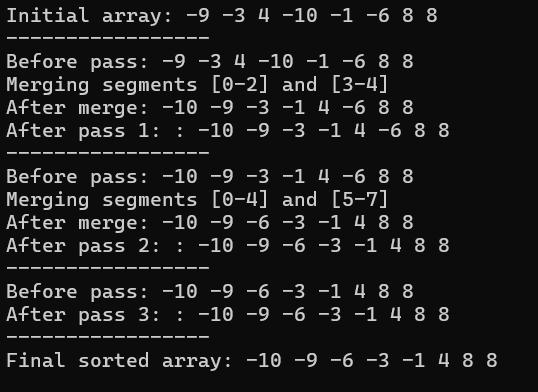
print\_array(a, 8, "Sorted array ");

return 0;

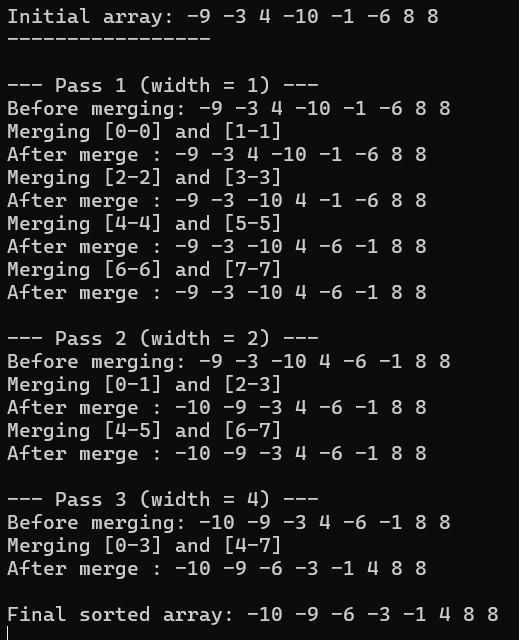
}

**Результат работы программ:**

1. Естественное слияние



1. Сбалансированное слияние



1. Многофазная

