

Практическая работа №2

Алгоритмы сортировки

Цель: Изучить алгоритмы сортировки.

Оснащение:

Персональный компьютер под управлением ОС Windows 10.

Офисный пакет Microsoft Office 2016

Среда разработки Visual Studio 2019

Ход работы

1. Пузырьковая сортировка

Сортировка пузырьком — один из самых известных алгоритмов сортировки. Здесь нужно последовательно сравнивать значения соседних элементов и менять числа местами, если предыдущее оказывается больше последующего. Таким образом, элементы с большими значениями оказываются в конце списка, а с меньшими остаются в начале.

					<i>АиСД.09.03.02.050000 ПР</i>		
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>			
<i>Разраб.</i>		<i>Ермошина В.А</i>			Лабораторная работа 2 Алгоритмы сортировки	<i>Лит.</i>	<i>Лист</i>
<i>Провер.</i>		<i>Берёза А.Н.</i>					
<i>Реценз</i>						2	10
<i>Н. Контр.</i>						<i>ИСОиП (филиал) ДГТУ в г.Шахты ИСТ-Тб21</i>	
<i>Утверд.</i>							

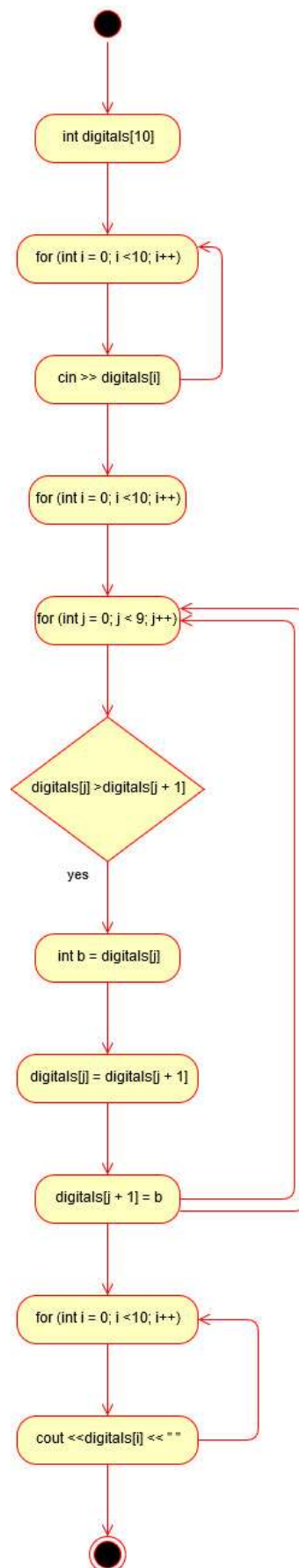


Рисунок 1 - Диаграмма деятельности пузырьковая сортировка

Код программы

#include <iostream>

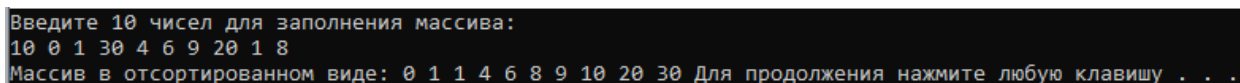
					<i>АуСД.09.03.02.050000 ПР</i>	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		

```

using namespace std;
int main() {
    setlocale(LC_ALL, "rus");
    int digital[10]; // объявили массив на 10 ячеек
    cout << "Введите 10 чисел для заполнения массива: " << endl;
    for (int i = 0; i < 10; i++) {
        cin >> digital[i]; // "читаем" элементы в массив
    }
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 9; j++) {
            if (digital[j] > digital[j + 1]) {
                int b = digital[j]; // создали дополнительную переменную
                digital[j] = digital[j + 1]; // меняем местами
                digital[j + 1] = b; // значения элементов
            }
        }
    }
    cout << "Массив в отсортированном виде: ";
    for (int i = 0; i < 10; i++) {
        cout << digital[i] << " "; // выводим элементы массива
    }
    system("pause");
    return 0;
}

```

Результат



Введите 10 чисел для заполнения массива:
10 0 1 30 4 6 9 20 1 8
Массив в отсортированном виде: 0 1 1 4 6 8 9 10 20 30 Для продолжения нажмите любую клавишу . . .

Рисунок 2 - Результат работы программы

2. Слияние массивов

Сортируемый массив разбивается на две части примерно одинакового размера. Каждая из получившихся частей сортируется отдельно, например, тем же самым алгоритмом. Два упорядоченных массива половинного размера соединяются в один.

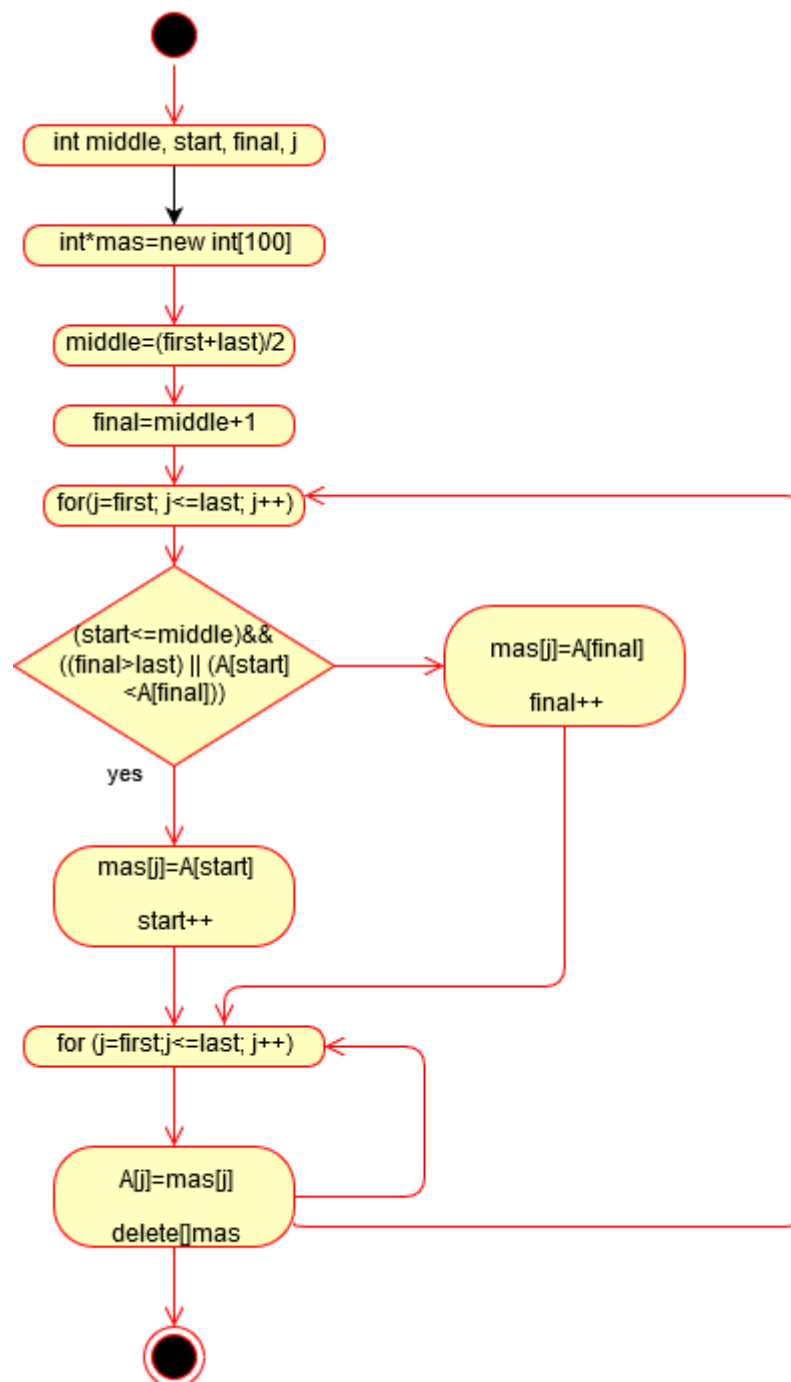


Рисунок 3 – Диаграмма деятельности сортировки слиянием

Код программы

```

#include <iostream>
using namespace std;
//функция, сливающая массивы
void Merge(int *A, int first, int last)
{
    int middle, start, final, j;
    int *mas=new int[100];
    middle=(first+last)/2; //вычисление среднего элемента
    start=first; //начало левой части
    final=middle+1; //начало правой части
    for(j=first; j<=last; j++) //выполнять от начала до конца
    if ((start<=middle) && ((final>last) || (A[start]<A[final])))

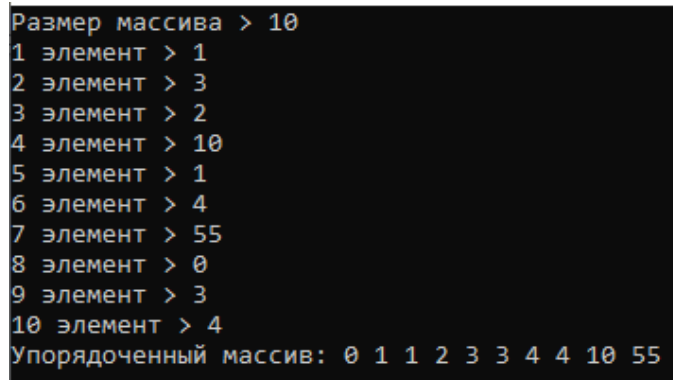
```

```

{
mas[j]=A[start];
start++;
}
else
{
mas[j]=A[final];
final++;
}
//возвращение результата в список
for (j=first; j<=last; j++) A[j]=mas[j];
delete[]mas;
};
//рекурсивная процедура сортировки
void MergeSort(int *A, int first, int last)
{
{
if (first<last)
{
MergeSort(A, first, (first+last)/2); //сортировка левой части
MergeSort(A, (first+last)/2+1, last); //сортировка правой части
Merge(A, first, last); //слияние двух частей
}
}
};
//главная функция
void main()
{
setlocale(LC_ALL, "Rus");
int i, n;
int *A=new int[100];
cout<<"Размер массива > "; cin>>n;
for (i=1; i<=n; i++)
{ cout<<i<<" элемент > "; cin>>A[i];}
MergeSort(A, 1, n); //вызов сортирующей процедуры
cout<<"Упорядоченный массив: "; //вывод упорядоченного массива
for (i=1; i<=n; i++) cout<<A[i]<<" ";
delete []A;
system("pause>>void");
}

```

Результат



```

Размер массива > 10
1 элемент > 1
2 элемент > 3
3 элемент > 2
4 элемент > 10
5 элемент > 1
6 элемент > 4
7 элемент > 55
8 элемент > 0
9 элемент > 3
10 элемент > 4
Упорядоченный массив: 0 1 1 2 3 3 4 4 10 55

```

Рисунок 4 – Результат работы программы

					<i>АИСД.09.03.02.050000 ПР</i>	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

3. Шейкерная сортировка

Шейкерная сортировка отличается от пузырьковой тем, что она двунаправленная: алгоритм перемещается не строго слева направо, а сначала слева направо, затем справа налево.

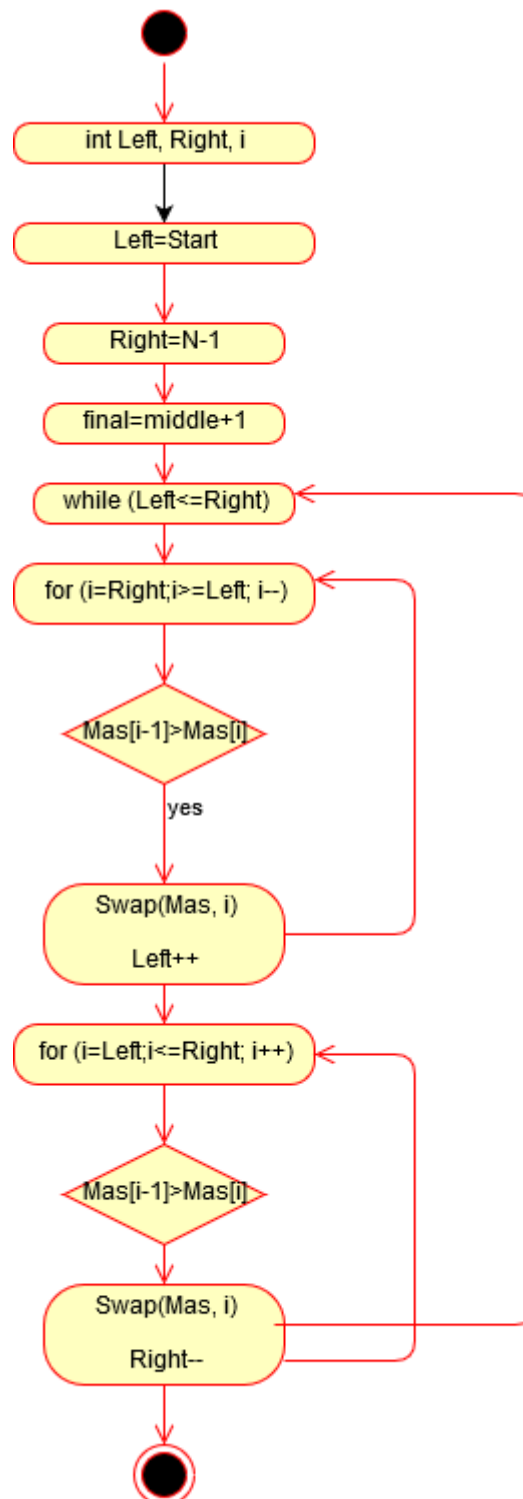


Рисунок 5 –Диаграмма деятельности алгоритм шейкерной сортировки

Код программы

```
#include <iostream>
using namespace std;
```

					AuCD.09.03.02.050000 ПР	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

```

//функция обмена
void Swap(int *Mas, int i)
{
    int temp;
    temp=Mas[i];
    Mas[i]=Mas[i-1];
    Mas[i-1]=temp;
}
//функция шейкерной сортировки
void ShakerSort(int *Mas, int Start, int N)
{
    int Left, Right, i;
    Left=Start;
    Right=N-1;
    while (Left<=Right)
    {
        for (i=Right; i>=Left; i--)
            if (Mas[i-1]>Mas[i]) Swap(Mas, i);
        Left++;
        for (i=Left; i<=Right; i++)
            if (Mas[i-1]>Mas[i]) Swap(Mas, i);
        Right--;
    }
}
//главная функция
void main()
{
    setlocale(LC_ALL,"Rus");
    int n, k;
    cout<<"Размер массива > "; cin>>n;
    int *A=new int[n];
    for (k=0; k<n; k++)
        { cout<<k+1<<" элемент > "; cin>>A[k]; }
    ShakerSort(A, 1, n);
    cout<<"Результирующий массив: ";
    for (k=0; k<n; k++)cout<<" "<<A[k];
    system("pause>>void");
}

```

Результат

```

Размер массива > 10
1 элемент > 2
2 элемент > 10
3 элемент > 1
4 элемент > 10
5 элемент > 3
6 элемент > 4
7 элемент > 0
8 элемент > 0
9 элемент > 2
10 элемент > 20
Результирующий массив:  0 0 1 2 2 3 4 10 10 20

```

Рисунок 6 – Результат работы программы

4. Сортировка Шелла

					<i>АИСД.09.03.02.050000 ПР</i>	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

Идея метода заключается в сравнение разделенных на группы элементов последовательности, находящихся друг от друга на некотором расстоянии. Изначально это расстояние равно d или $N/2$, где N — общее число элементов. На первом шаге каждая группа включает в себя два элемента расположенных друг от друга на расстоянии $N/2$; они сравниваются между собой, и, в случае необходимости, меняются местами. На последующих шагах также происходят проверка и обмен, но расстояние d сокращается на $d/2$, и количество групп, соответственно, уменьшается. Постепенно расстояние между элементами уменьшается, и на $d=1$ проход по массиву происходит в последний раз.

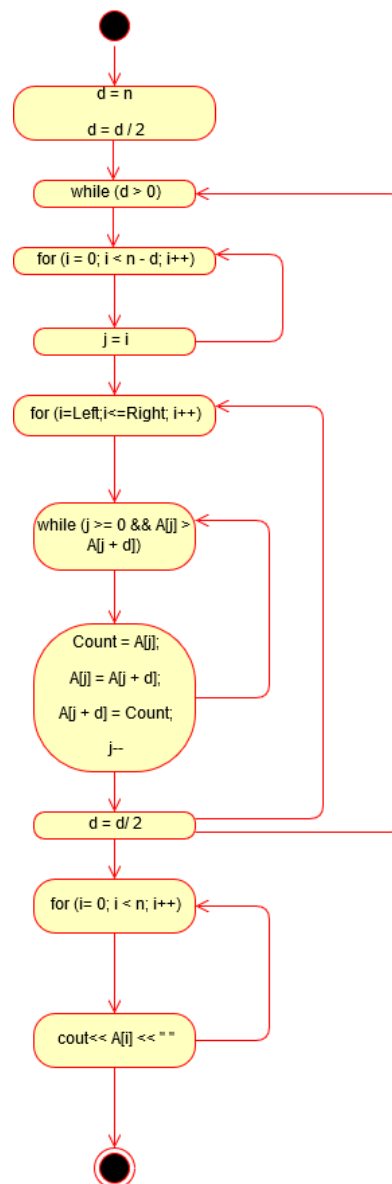
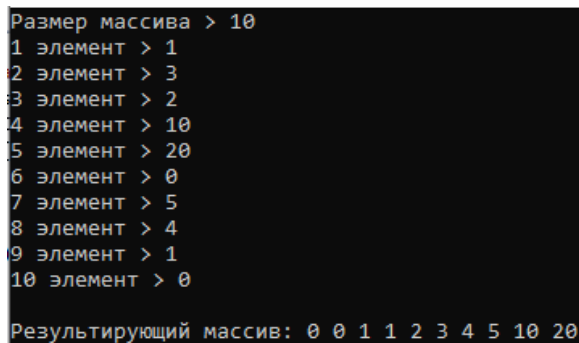


Рисунок 7 – Диаграмма деятельности алгоритм сортировки Шелла

Код программы

```
#include <iostream>
using namespace std;
int i, j, n, d, Count;
void Shell(int A[], int n) //сортировка Шелла
{
    d = n;
    d = d / 2;
    while (d > 0)
    {
        for (i = 0; i < n - d; i++)
        {
            j = i;
            while (j >= 0 && A[j] > A[j + d])
            {
                Count = A[j];
                A[j] = A[j + d];
                A[j + d] = Count;
                j--;
            }
        }
        d = d / 2;
    }
    for (i = 0; i < n; i++) cout << A[i] << " "; //вывод массива
}
//главная функция
void main()
{
    setlocale(LC_ALL, "Rus");
    cout << "Размер массива > "; cin >> n;
    int *A = new int[n]; //объявление динамического массива
    for (i = 0; i < n; i++) //ввод массива
    {
        cout << i + 1 << " элемент > "; cin >> A[i];
    }
    cout << "\nРезультирующий массив: ";
    Shell(A, n);
    delete[] A; //освобождение памяти
    system("pause>>void");
}
```

Результат



```
Размер массива > 10
1 элемент > 1
2 элемент > 3
3 элемент > 2
4 элемент > 10
5 элемент > 20
6 элемент > 0
7 элемент > 5
8 элемент > 4
9 элемент > 1
10 элемент > 0

Результирующий массив: 0 0 1 1 2 3 4 5 10 20
```

Рисунок 8 - Результат работы программы

					АИСД.09.03.02.050000 ПР	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

Быстрая сортировка

Быстрая сортировка — это алгоритм сортировки, сложность которого в среднем составляет $O(n \log(n))$. Суть его предельно проста: выбирается так называемый опорный элемент, и массив делится на 3 подмассива: меньших опорного, равных опорному и больших опорного. Потом этот алгоритм применяется рекурсивно к подмассивам.

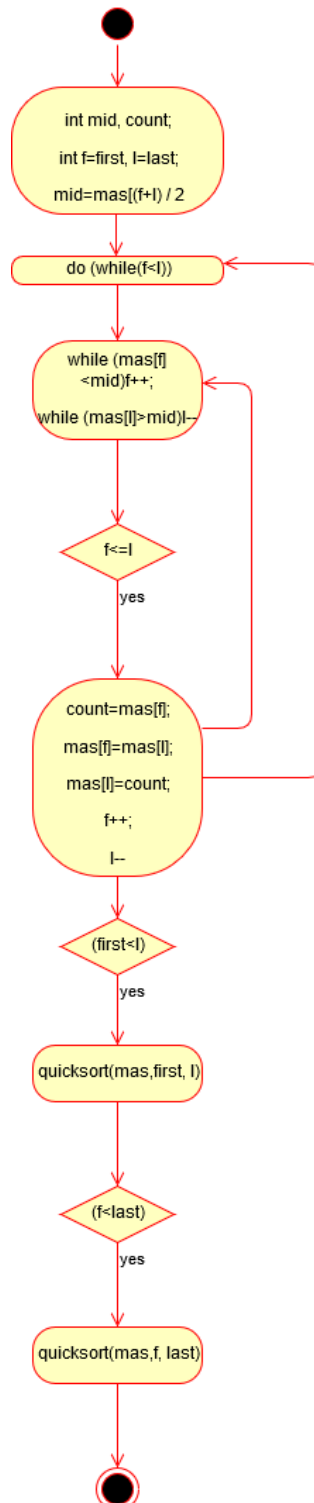
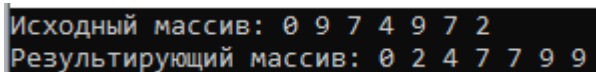


Рисунок 9 – Диаграмма деятельности быстрой сортировки

Код программы

```
#include <iostream>
#include <ctime>
using namespace std;
const int n=7;
int first, last;
//функция сортировки
void quicksort(int *mas, int first, int last)
{
    int mid, count;
    int f=first, l=last;
    mid=mas[(f+l) / 2]; //вычисление опорного элемента
    do
    {
        while (mas[f]<mid) f++;
        while (mas[l]>mid) l--;
        if (f<=l) //перестановка элементов
        {
            count=mas[f];
            mas[f]=mas[l];
            mas[l]=count;
            f++;
            l--;
        }
    } while (f<l);
    if (first<l) quicksort(mas, first, l);
    if (f<last) quicksort(mas, f, last);
}
//главная функция
void main()
{
    setlocale(LC_ALL,"Rus");
    int *A=new int[n];
    srand(time(NULL));
    cout<<"Исходный массив: ";
    for (int i=0; i<n; i++)
    {
        A[i]=rand()%10;
        cout<<A[i]<<" ";
    }
    first=0; last=n-1;
    quicksort(A, first, last);
    cout<<endl<<"Результирующий массив: ";
    for (int i=0; i<n; i++) cout<<A[i]<<" ";
    delete []A;
    system("pause>>void");
}
```

Результат



```
Исходный массив: 0 9 7 4 9 7 2
Результирующий массив: 0 2 4 7 7 9 9
```

Рисунок 10 - Результат работы программы

					АИСД.09.03.02.050000 ПР	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

Вывод: В ходе выполнения практической работы изучила алгоритмы сортировки и написала программы, реализующие пузырьковую сортировку, сортировку слиянием, шейкерную сортировку, сортировку Шелла и быструю сортировку.

					<i>АиСД.09.03.02.050000 ПР</i>	<i>Лист</i>
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		<i>13</i>