МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
Київський національний університет імені Тараса Шевченка
Кафедра програмних систем і технологій

Звіт з лабораторної роботи 2.3

Тема: «Сортування масивів 3д»

Виконав: студент групи ІПЗ - 12
Богатько Олександр
Геннадійович
Перевірила: викладач
Юрчук Ірина Аркадіївна

Київ 2020

Мета: вивчити сортування масивів, використовуючи алгоритми.

1.Алгоритм пірамідального сортування(heap sort).

Код программи:

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

void heapify(int arr[], int n, int i)
{
    int largest = i;
    int l = 2*i + 1;
    int r = 2*i + 2;
    if (l < n && arr[l] > arr[largest])
        largest = l;

    if (r < n && arr[r] > arr[largest])
        largest = r;

    if (largest != i)
    {
        swap(arr[i], arr[largest]);
        heapify(arr, n, largest);
    }
}



void printArray(int arr[], int n)
{
    for (int i=0; i<n; ++i)
        cout << arr[i] << " ";
    cout << "\n";
}
void heapSort(int arr[], int n)
{   int reps = 0; int checks = 0;
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);
    for (int i=n-1; i>0; i--)
    {
        swap(arr[0], arr[i]);
        heapify(arr, i, 0);
        cout<<"\nInteration ["<<reps<<"]: ";
        printArray(arr, n);
        reps++;
    }
cout<<"\nNumber of replaces: "<< reps<<"\n";
}

int main()
{
    int n;
    char mode;
    cout<<"Enter size: ";
    cin>>n;
    int arr[n];
    cout<<"1.Random fill\n"
          "2.Manual fill\n"
          "Choose mode: ";
    cin>>mode;
    switch(mode){

        case'1':
            for(int i = 0; i < n; ++i) arr[i] = rand()%100;
            cout << "Generated array: ";
            printArray(arr, n);
            heapSort(arr,n);
            cout << "\nSorted array is ";
            printArray(arr, n);
            cout << endl;
            break;

        case'2':
            for(int i = 0;i<n;i++) {
```

```
                cout << "\nEnter element: ";    //показать итерации
                cin >> arr[i];
            }
            heapSort(arr,n);
            cout << "Sorted array is: \n";
            printArray(arr, n);
            cout<<endl;
            break;

        default:break;
    }
}
```

Результати роботи програми:

```
Enter size: 10
1.Random fill
2.Manual fill
Choose mode: 1
Generated array: 7 49 73 58 30 72 44 78 23 9

Interation [0]: 73 58 72 49 30 9 44 7 23 78

Interation [1]: 72 58 44 49 30 9 23 7 73 78

Interation [2]: 58 49 44 7 30 9 23 72 73 78

Interation [3]: 49 30 44 7 23 9 58 72 73 78

Interation [4]: 44 30 9 7 23 49 58 72 73 78

Interation [5]: 30 23 9 7 44 49 58 72 73 78

Interation [6]: 23 7 9 30 44 49 58 72 73 78

Interation [7]: 9 7 23 30 44 49 58 72 73 78

Interation [8]: 7 9 23 30 44 49 58 72 73 78

Number of replaces: 9
Sorted array is 7 9 23 30 44 49 58 72 73 78


Process finished with exit code 0
```

```
Enter size: 10
1.Random fill
2.Manual fill
Choose mode: 2

Enter element: 23

Enter element: 12

Enter element: 52

Enter element: 58

Enter element: 97

Enter element: 66

Enter element: 44

Enter element: 35

Enter element: 91

Enter element: 84

Interation [0]: 91 84 66 58 12 52 44 35 23 97

Interation [1]: 84 58 66 35 12 52 44 23 91 97

Interation [2]: 66 58 52 35 12 23 44 84 91 97

Interation [3]: 58 44 52 35 12 23 66 84 91 97

Interation [4]: 52 44 23 35 12 58 66 84 91 97

Interation [5]: 44 35 23 12 52 58 66 84 91 97

Interation [6]: 35 12 23 44 52 58 66 84 91 97

Interation [7]: 23 12 35 44 52 58 66 84 91 97

Interation [8]: 12 23 35 44 52 58 66 84 91 97

Number of replaces: 9
Sorted array is:
12 23 35 44 52 58 66 84 91 97


Process finished with exit code 0
```

2.Алгоритм паралельного сортування Батчера(обмінне сортування зі злиттям).

Код программи:

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

void printArray(int arr[], int n)
{
    for (int i=0; i<n; ++i)
        cout << arr[i] << " ";
    cout << "\n";
}
template<typename T>
void MergeSort(T a[], size_t l)
{
    int checks = 0;
    int merge = 0;

    size_t BlockSizeIterator;
    size_t BlockIterator;
    size_t LeftBlockIterator;
    size_t RightBlockIterator;
    size_t MergeIterator;

    size_t LeftBorder;
    size_t MidBorder;
    size_t RightBorder;
    for (BlockSizeIterator = 1; BlockSizeIterator < l; BlockSizeIterator *= 2)
    {
        for (BlockIterator = 0; BlockIterator < l - BlockSizeIterator; BlockIterator += 2 *
BlockSizeIterator)
        {
            merge++;
            LeftBlockIterator = 0;
            RightBlockIterator = 0;
            LeftBorder = BlockIterator;
            MidBorder = BlockIterator + BlockSizeIterator;
            RightBorder = BlockIterator + 2 * BlockSizeIterator;
            RightBorder = (RightBorder < l) ? RightBorder : l;
            int* SortedBlock = new int[RightBorder - LeftBorder];

            while (LeftBorder + LeftBlockIterator < MidBorder && MidBorder + RightBlockIterator <
RightBorder)
            {
                if (a[LeftBorder + LeftBlockIterator] < a[MidBorder + RightBlockIterator])
                {
                    SortedBlock[LeftBlockIterator + RightBlockIterator] = a[LeftBorder +
LeftBlockIterator];
                    LeftBlockIterator += 1;
                }
                else
                {
                    SortedBlock[LeftBlockIterator + RightBlockIterator] = a[MidBorder +
RightBlockIterator];
                    RightBlockIterator += 1;
                }
            }
            while (LeftBorder + LeftBlockIterator < MidBorder)
            {
                SortedBlock[LeftBlockIterator + RightBlockIterator] = a[LeftBorder +
LeftBlockIterator];
                LeftBlockIterator += 1;
            }
            while (MidBorder + RightBlockIterator < RightBorder)
            {
                SortedBlock[LeftBlockIterator + RightBlockIterator] = a[MidBorder +
RightBlockIterator];
                RightBlockIterator += 1;
            }

            for (MergeIterator = 0; MergeIterator < LeftBlockIterator + RightBlockIterator;
MergeIterator++)
            {
                a[LeftBorder + MergeIterator] = SortedBlock[MergeIterator];
```

```
            }
            delete SortedBlock;
        }
        cout<<"\nIteration ["<<checks<< "]: ";
        printArray(a,l);
        checks++;
    }
    cout<<"\nNumber of merges: "<< merge;
}
int main() {
    int n;
    char mode;
    cout << "Enter size: ";
    cin >> n;
    int arr[n];
    cout << "1.Random fill\n"
            "2.Manual fill\n"
            "Choose mode: ";
    cin >> mode;
    switch (mode) {

        case '1':
            for (int i = 0; i < n; ++i) arr[i] = rand() % 100;
            cout << "Generated array: ";
            printArray(arr, n);
            break;

        case '2':
            for (int i = 0; i < n; i++) {
                cout << "\nEnter element: ";    //показать итерации
                cin >> arr[i];
            }

        default:
            break;
    }
    MergeSort(arr, n); //сортировка
    cout << "\nSorted array is ";
    printArray(arr, n);
    cout << endl;
```

Результати роботи програми:

```
Enter size: 10
1.Random fill
2.Manual fill
Choose mode: 1
Generated array: 7 49 73 58 30 72 44 78 23 9

Iteration [0]: 7 49 58 73 30 72 44 78 9 23

Iteration [1]: 7 49 58 73 30 44 72 78 9 23

Iteration [2]: 7 30 44 49 58 72 73 78 9 23

Iteration [3]: 7 9 23 30 44 49 58 72 73 78

Number of merges: 9
Sorted array is 7 9 23 30 44 49 58 72 73 78


Process finished with exit code 0
```

```
Enter size: 10
1.Random fill
2.Manual fill
Choose mode: 2

Enter element: 41

Enter element: 24

Enter element: 35

Enter element: 57

Enter element: 85

Enter element: 99

Enter element: 73

Enter element: 12

Enter element: 29

Enter element: 64

Iteration [0]: 24 41 35 57 85 99 12 73 29 64

Iteration [1]: 24 35 41 57 12 73 85 99 29 64

Iteration [2]: 12 24 35 41 57 73 85 99 29 64

Iteration [3]: 12 24 29 35 41 57 64 73 85 99

Number of merges: 9
Sorted array is 12 24 29 35 41 57 64 73 85 99


Process finished with exit code 0
```