Министерство образования Республики Беларусь
Учреждение образования «Полоцкий государственный университет»

Кафедра технологий программирования

Основы алгоритмизации и программирования

Отчет по лабораторной работе №9

Вариант 11

Выполнил — Ланцев Евгений Николаевич. 21-ИТ-1, ФИТ

Проверил — Пантелейко А.Ф. Преподаватель-стажер

Полоцк
2022 г.

Лабораторная работа № 9

“Сортировка и поиск”

Цель работы: изучить быстрые и медленные алгоритмы сортировки. Изучить алгоритмы линейного и бинарного поиска.

Вариант 11

“Очередь типа int”

```
struct node{
    int data;
    node *next;
    node(int p_data,node *p_next){
        data = p_data; next = p_next;
    }
};
```

Рисунок 1 - Структура узла.

```
struct l{
    node *root;
    l(int data){
        root = new node(data, nullptr);
    }
```

Рисунок 2 - Структура очереди.

```
void insert_node(int data){
    if(!root){
        root = new node(data,nullptr);
    }else {
        node* last = get_last(root);
        node* new_node = new node(data,nullptr);
        last→next = new_node;
    }
}
```

Рисунок 3 - Методы добавления узла в очередь.

```
void remove_node(){
    node* new_root = root→next;
    delete root;
    root = new_root;
}
```

Рисунок 4 - Методы удаления узла из очереди.

```
void print_l(){ // Вывод всех элементов
    node * temp = root;
    while(temp){
        std::cout << temp→data << " "; temp = temp→next;
    }
}
```

Рисунок 5 - Метод вывода узлов из очереди.

```
void done_quene(){
    while(root){
        remove_node();
    }
    delete root;
}
```

Рисунок 6 - Метод удаления всех узлов из очереди.

```
int load_to_array(node *n[],l *quene,int length_nodes){
    std::cout << "Loading to a dynamic array...\n";
    for(node *i = quene->root;i != nullptr;i = i->next){
        n[length_nodes] = new node(i->data,i->next);
        length_nodes++;
    }
    std::cout << "Done!\n";
    return length_nodes;
}

void load_to_struct(node *n[],l *quene,int length_nodes){
    std::cout << "Loading to a struct...\n";
    for(int i = 0; i < length_nodes;i++){
        //std::cout << nodes[i]->data << " ";
        quene->insert_node(n[i]->data);
    }
    std::cout << "Done!\n";
}
```

Рисунок 7 - Методы загрузки и выгрузки данных из структуры в динамический массив.

```
void swap(node *n[],int x,int y){
    node *temp = n[y];
    n[y] = n[x];
    n[x] = temp;
}

void sort_array_bublle(node *n[], int length_array){ // сортировка масиива
    std::cout << "Sorting array...\n";
    for(int i = 0;i < length_array; i++){ // каждый узел сравниваем с каждым узлом
        for(int k = 0; k < length_array - i - 1;k++){
            if(n[k]->data > n[k+1]->data){ // если узел который меньше другога узла стоит позже
                swap(n,k,k+1);
            }
        }
    }
}
```

Рисунок 8 - Метод сортировки пузырьком.

```
int linear_search(node *n[],int value,int length_array,int start){
    for(int i = start;i < length_array;i++){
        if(n[i]->data == value){
            return i;
        }
    }
    return -1;
}

int linear_double_search(node *n[],int value,int length_array){ // проверяе
    int count = 0;
    int ind = -1;
    while((ind = linear_search(n,value,length_array,ind + 1)) != -1){
        count++;
    }
    return count;
}
```

Рисунок 9 - Метод двоичного поиска.

```
int length_nodes = 0;
int le;
std::cout << "Input a length of array -> ";
std::cin >> le;
const int y = le;
node *nodes[y]; // динамический массив узлов
```

Рисунок 10 - Создания динамического массива.

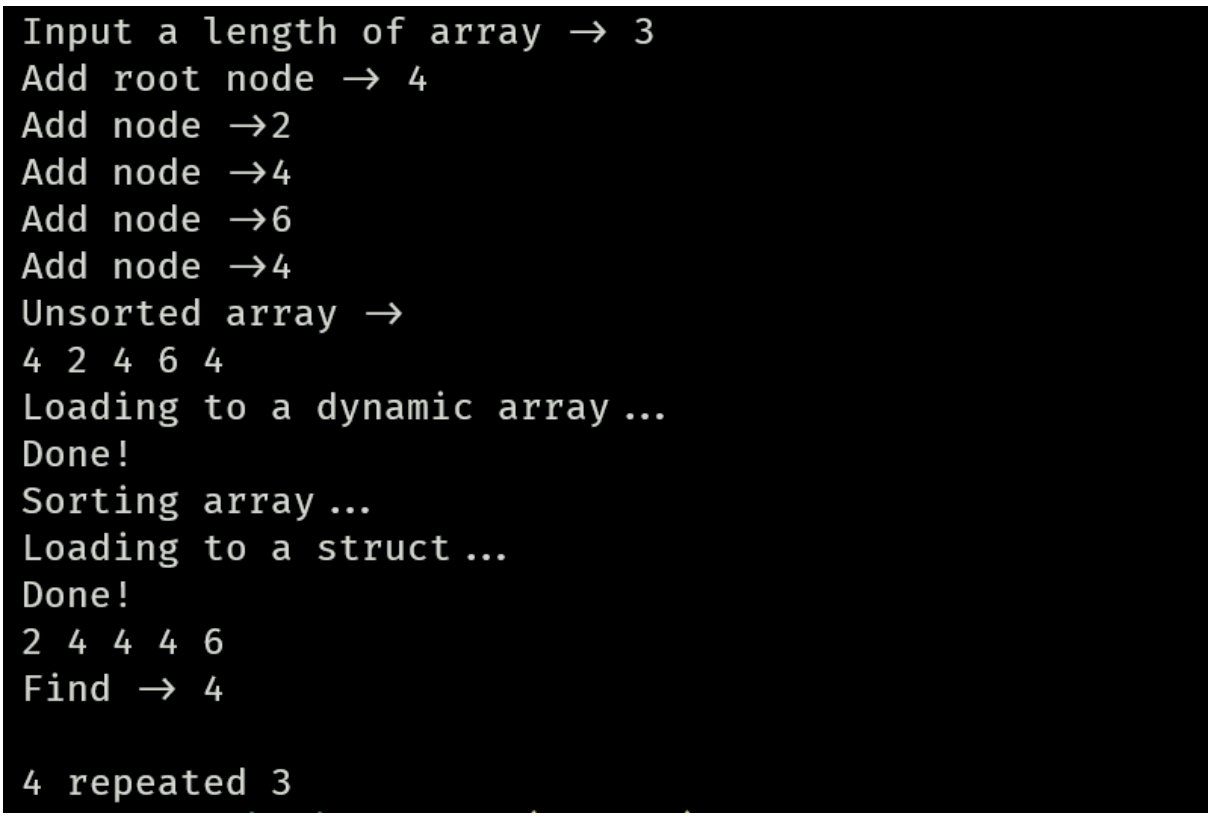

Рисунок 11 - Работа программы.

Вывод:Я изучить быстрые и медленные алгоритмы сортировки. Изучить алгоритмы линейного и бинарного поиска.