**Пензенский государственный университет**

**Кафедра вычислительной техники**

**Отчет**

по лабораторной работе № 3

по дисциплине: “Логика и алгоритмизация в инженерных задачах"

Выполнили студенты группы 22ВВП2:

Бормотов А.А.

Кузьмин Д.В.

Кочетков А.М.

Приняли:

Юрова О.В.

Акифьев И.В.

2023

**Цель**: Реализовать динамические списки в виде приоритетной очереди, стэка.

**Задание 1**

Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <fstream>

#include <iostream>

#include <cstdlib>

using namespace std;

struct Element //creating structure for all elements of list

{

char name[20];

char value[10];

int priority;

Element\* next;

};

class Dynamic\_List //creating class for dynamic lists

{

public:

Element\* head; //pointer for head of list

Element\* tail; //pointer for tail of list

Dynamic\_List() //constructor for all Dynamic\_List objects

{

head = NULL; // => empty list

tail = NULL;

}

bool Check\_Empty()

{

return (head == NULL);

}

void Element\_in(char\* Name, char\* Value, char\* Priority) //function for adding elements in list

{

Element\* movable\_var = new Element();

movable\_var = head;

Element\* element = new Element();

Element\* prev\_var = new Element();

prev\_var = head;

strcpy\_s(element->name, 20, Name); //reading info from file

strcpy\_s(element->value, 10, Value);

element->priority = atoi(Priority); //

element->next = NULL;

if (Check\_Empty())

{

head = element;

tail = element; //if list is empty place element at the head

}

else

{

if (element->priority > head->priority)

{

element->next = head;

head = element;

return;

}

while (movable\_var != NULL)

{

if (element->priority > movable\_var->priority)

{

prev\_var->next = element;

element->next = movable\_var;

return;

}

prev\_var = movable\_var;

movable\_var = movable\_var->next;

}

if (element->next == NULL)

{

tail = element;

prev\_var->next = tail;

}

}

}

Element\* Search(char\* Name) //function for searching for element by name

{

Element\* movable\_var = new Element(); //if desired element not first, look through list with additional variable

movable\_var = head;

while (movable\_var != NULL)

{

if (!strcmp(movable\_var->name, Name))

{

return movable\_var;

}

movable\_var = movable\_var->next;

}

return movable\_var;

}

void Element\_Remove(Element\* element) //function for removing element from list

{

if (element == head) //if desired element is first, move pointer forward

{

head = element->next;

delete element;

return;

}

Element\* movable\_var = new Element(); //if desired element not first, look through list with additional variable

movable\_var = head;

while (movable\_var->next != element)

{

movable\_var = movable\_var->next;

}

movable\_var->next = element->next;

delete element;

}

void Element\_Edit(Element& element, char\* Value) //function for editing desired element's value

{

strcpy\_s(element.value, 10, Value);

}

void Element\_Remove()

{

head = head->next;

}

void Queue\_Display()

{

Element\* movable\_var = new Element();

movable\_var = head;

while (movable\_var != NULL) {

cout << movable\_var->name << " ";

cout << movable\_var->value << " ";

cout << movable\_var->priority << " " << endl;

movable\_var = movable\_var->next;

}

cout << "----------------\n";

}

};

int main(clock\_t end\_s)

{

char\* FileName = new char[50];

char\* buf\_name = new char[20];

char\* buf\_value = new char[10];

char\* buf\_priority = new char[3];

char\* search\_elem = new char[20];

char\* new\_value = new char[10];

char choice = ' ';

Dynamic\_List myList; //creating dynamic list

cout << "Enter name of file: ";

cin >> FileName;

ifstream\* inPointer = new ifstream(FileName);

if (!inPointer->good())

{

cout << "Error opening file!\n";

system("PAUSE");

return 0;

}

while (!inPointer->eof()) //reading file data

{

inPointer->getline(buf\_name, 20, ' ');

inPointer->getline(buf\_value, 10, ' ');

inPointer->getline(buf\_priority, 3, ' ');

myList.Element\_in(buf\_name, buf\_value, buf\_priority);

}

inPointer->close();

cout << "Current queue:\n";

myList.Queue\_Display();

while (choice != '9')

{

cout << "What action to perfrom:\n 1 - add new element\n 2 - delete element\n 3 - delete element by value\n 4 - edit element\n 9 - exit programm\n ";

cin >> choice;

switch (choice)

{

case '1':

{

cout << "Enter data of new element that will be added to the list: ";

cin >> buf\_name;

cout << "Enter value: ";

cin >> buf\_value;

cout << "And the priority of a new element: ";

cin >> buf\_priority;

myList.Element\_in(buf\_name, buf\_value, buf\_priority);

myList.Queue\_Display();

break;

}

case '2':

{

myList.Element\_Remove();

myList.Queue\_Display();

break;

}

case '3':

{

cout << "Enter element to remove: ";

cin >> search\_elem;

if (myList.Search(search\_elem)) //removing element from list

{

myList.Element\_Remove(myList.Search(search\_elem));

}

myList.Queue\_Display();

break;

}

case '4':

{

cout << "Enter element to edit: ";

cin >> search\_elem;

cout << "Enter new value: ";

cin >> new\_value;

if (myList.Search(search\_elem)) //editing element's value

{

myList.Element\_Edit(\*myList.Search(search\_elem), new\_value);

}

myList.Queue\_Display();

break;

}

case '9':

{

cout << "Exiting programm... \n";

break;

}

default:

cout << "NUH-UH!";

break;

}

}

ofstream\* out = new ofstream(FileName); //output for final list in file

while (myList.head != NULL)

{

out->write(myList.head->name, strlen(myList.head->name));

out->write(" ", 1);

out->write(myList.head->value, strlen(myList.head->value));

out->write(" ", 1);

\_itoa(myList.head->priority, buf\_priority, 10);

out->write(buf\_priority, strlen(buf\_priority));

out->write(" ", 1);

myList.head = myList.head->next;

}

out->close();

system("PAUSE");

return 0;

}

**Задание 2**

На основе приведенного кода реализовать структуру данных Очередь.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <fstream>

#include <iostream>

#include <cstdlib>

using namespace std;

struct Element //creating structure for all elements of list

{

char name[20];

char value[10];

Element\* next;

};

class Dynamic\_List //creating class for dynamic lists

{

public:

Element\* head; //pointer for head of list

Element\* tail; //pointer for tail of list

Dynamic\_List() //constructor for all Dynamic\_List objects

{

head = NULL; // => empty list

}

bool Check\_Empty()

{

return (head == NULL);

}

void Element\_in(char\* Name, char\* Value) //function for adding elements in list

{

Element\* element = new Element(); //allocating memory for new Element object

strcpy\_s(element->name, 20, Name); //reading info from file

strcpy\_s(element->value, 10, Value);

element->next = NULL;

if (Check\_Empty())

{

head = element; //if list is empty place element at the head

}

else

{

tail->next = element; //if list had atleast 1 element already place new element at the tail

}

tail = element;

}

Element\* Search(char\* Name) //function for searching for element by name

{

Element\* movable\_var = new Element(); //if desired element not first, look through list with additional variable

movable\_var = head;

while (movable\_var != NULL)

{

if (!strcmp(movable\_var->name, Name))

{

return movable\_var;

}

movable\_var = movable\_var->next;

}

return movable\_var;

}

void Element\_Remove(Element\* element) //function for removing element from list

{

if (element == head) //if desired element is first, move pointer forward

{

head = element->next;

delete element;

return;

}

Element\* movable\_var = new Element(); //if desired element not first, look through list with additional variable

movable\_var = head;

while (movable\_var->next != element)

{

movable\_var = movable\_var->next;

}

movable\_var->next = element->next;

delete element;

}

void Element\_Edit(Element& element, char\* Value) //function for editing desired element's value

{

strcpy\_s(element.value, 10, Value);

}

void Element\_Remove()

{

head = head->next;

}

void Queue\_Display()

{

Element\* movable\_var = new Element();

movable\_var = head;

while (movable\_var != NULL) {

cout << movable\_var->name << " ";

cout << movable\_var->value << " " << endl;

movable\_var = movable\_var->next;

}

cout << "----------------\n";

}

};

int main(clock\_t end\_s)

{

char\* FileName = new char[50];

char\* buf\_name = new char[20];

char\* buf\_value = new char[10];

char\* search\_elem = new char[20];

char\* new\_value = new char[10];

char choice = ' ';

Dynamic\_List myList; //creating dynamic list

cout << "Enter name of file: ";

cin >> FileName;

ifstream\* inPointer = new ifstream(FileName);

if (!inPointer->good())

{

cout << "Error opening file!\n";

system("PAUSE");

return 0;

}

while (!inPointer->eof()) //reading file data

{

inPointer->getline(buf\_name, 20, ' ');

inPointer->getline(buf\_value, 10, ' ');

myList.Element\_in(buf\_name, buf\_value);

}

inPointer->close();

cout << "Current queue:\n";

myList.Queue\_Display();

while (choice != '9')

{

cout << "What action to perfrom:\n 1 - add new element\n 2 - delete element\n 3 - delete element by value\n 4 - edit element\n 9 - exit programm\n ";

cin >> choice;

switch (choice)

{

case '1':

{

cout << "Enter data of new element that will be added to the list: ";

cin >> buf\_name;

cout << "Enter value: ";

cin >> buf\_value;

myList.Element\_in(buf\_name, buf\_value);

myList.Queue\_Display();

break;

}

case '2':

{

myList.Element\_Remove();

myList.Queue\_Display();

break;

}

case '3':

{

cout << "Enter element to remove: ";

cin >> search\_elem;

if (myList.Search(search\_elem)) //removing element from list

{

myList.Element\_Remove(myList.Search(search\_elem));

}

myList.Queue\_Display();

break;

}

case '4':

{

cout << "Enter element to edit: ";

cin >> search\_elem;

cout << "Enter new value: ";

cin >> new\_value;

if (myList.Search(search\_elem)) //editing element's value

{

myList.Element\_Edit(\*myList.Search(search\_elem), new\_value);

}

myList.Queue\_Display();

break;

}

case '9':

{

cout << "Exiting programm... \n";

break;

}

default:

cout << "NUH-UH!";

break;

}

}

ofstream\* out = new ofstream(FileName); //output for final list in file

while (myList.head != NULL)

{

out->write(myList.head->name, strlen(myList.head->name));

out->write(" ", 1);

out->write(myList.head->value, strlen(myList.head->value));

out->write(" ", 1);

myList.head = myList.head->next;

}

out->close();

system("PAUSE");

return 0;

}

**Задание 3**

На основе приведенного кода реализовать структуру данных Стек.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <fstream>

#include <iostream>

#include <cstdlib>

using namespace std;

struct Element //creating structure for all elements of list

{

char name[20];

char value[10];

Element\* next;

};

class Dynamic\_List //creating class for dynamic lists

{

public:

Element\* head; //pointer for head of list

Element\* tail; //pointer for tail of list

Dynamic\_List() //constructor for all Dynamic\_List objects

{

head = NULL; // => empty list

}

bool Check\_Empty()

{

return (head == NULL);

}

void Element\_in(char\* Name, char\* Value) //function for adding elements in list

{

Element\* movable\_var = new Element();

movable\_var = head;

Element\* element = new Element();

Element\* prev\_head = new Element();

prev\_head = head;

strcpy\_s(element->name, 20, Name); //reading info from file

strcpy\_s(element->value, 10, Value); //

element->next = NULL;

if (Check\_Empty())

{

head = element;

tail = element; //if list is empty place element at the head

}

else

{

element->next = prev\_head;

head = element;

}

}

Element\* Search(char\* Name) //function for searching for element by name

{

Element\* movable\_var = new Element(); //if desired element not first, look through list with additional variable

movable\_var = head;

while (movable\_var != NULL)

{

if (!strcmp(movable\_var->name, Name))

{

return movable\_var;

}

movable\_var = movable\_var->next;

}

return movable\_var;

}

void Element\_Remove(Element\* element) //function for removing element from list

{

if (element == head) //if desired element is first, move pointer forward

{

head = element->next;

delete element;

return;

}

Element\* movable\_var = new Element(); //if desired element not first, look through list with additional variable

movable\_var = head;

while (movable\_var->next != element)

{

movable\_var = movable\_var->next;

}

movable\_var->next = element->next;

delete element;

}

void Element\_Edit(Element& element, char\* Value) //function for editing desired element's value

{

strcpy\_s(element.value, 10, Value);

}

void Element\_Remove()

{

head = head->next;

}

void Queue\_Display()

{

Element\* movable\_var = new Element();

movable\_var = head;

while (movable\_var != NULL) {

cout << movable\_var->name << " ";

cout << movable\_var->value << " " << endl;

movable\_var = movable\_var->next;

}

cout << "----------------\n";

}

};

int main(clock\_t end\_s)

{

char\* FileName = new char[50];

char\* buf\_name = new char[20];

char\* buf\_value = new char[10];

char\* search\_elem = new char[20];

char\* new\_value = new char[10];

char choice = ' ';

Dynamic\_List myList; //creating dynamic list

cout << "Enter name of file: ";

cin >> FileName;

ifstream\* inPointer = new ifstream(FileName);

if (!inPointer->good())

{

cout << "Error opening file!\n";

system("PAUSE");

return 0;

}

while (!inPointer->eof()) //reading file data

{

inPointer->getline(buf\_name, 20, ' ');

inPointer->getline(buf\_value, 10, ' ');

myList.Element\_in(buf\_name, buf\_value);

}

inPointer->close();

cout << "Current Stack:\n";

myList.Queue\_Display();

while (choice != '9')

{

cout << "What action to perfrom:\n 1 - add new element\n 2 - delete element\n 3 - delete element by value\n 4 - edit element\n 9 - exit programm\n ";

cin >> choice;

switch (choice)

{

case '1':

{

cout << "Enter data of new element that will be added to the list: ";

cin >> buf\_name;

cout << "Enter value: ";

cin >> buf\_value;

myList.Element\_in(buf\_name, buf\_value);

myList.Queue\_Display();

break;

}

case '2':

{

myList.Element\_Remove();

myList.Queue\_Display();

break;

}

case '3':

{

cout << "Enter element to remove: ";

cin >> search\_elem;

if (myList.Search(search\_elem)) //removing element from list

{

myList.Element\_Remove(myList.Search(search\_elem));

}

myList.Queue\_Display();

break;

}

case '4':

{

cout << "Enter element to edit: ";

cin >> search\_elem;

cout << "Enter new value: ";

cin >> new\_value;

if (myList.Search(search\_elem)) //editing element's value

{

myList.Element\_Edit(\*myList.Search(search\_elem), new\_value);

}

myList.Queue\_Display();

break;

}

case '9':

{

cout << "Exiting programm... \n";

break;

}

default:

cout << "NUH-UH!";

break;

}

}

ofstream\* out = new ofstream(FileName); //output for final list in file

while (myList.head != NULL)

{

out->write(myList.head->name, strlen(myList.head->name));

out->write(" ", 1);

out->write(myList.head->value, strlen(myList.head->value));

out->write(" ", 1);

myList.head = myList.head->next;

}

out->close();

system("PAUSE");

return 0;

}

**Вывод:** В результате выполнения лабораторной работы был модифицирован исходный код для реализации сложных структур данных, таких как: приоритетная и обычная очередь, стэк.  
 Получены опыт в реализации и работе с динамическими списками.