Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»
КАФЕДРА ИУК2 «Информационные системы и сети»

Домашняя работа
«Однонаправленный список. Стек. Очередь.»

ДИСЦИПЛИНА: «Объектно-ориентированное программирование»

Выполнил: студент гр. ИУК4-21Б (Суриков Н.С)
(подпись) (Ф.И.О.)

Проверил: (Дерюгина Е. О.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2024

Цель: реализовать класс для работы с однонаправленными списками.

Задачи:

1. Создание стека
2. Создание очереди
3. Проход по списку
4. Удаление из списка по ключу
5. Добавления элемента за ключевым в список

Листинг программы:

Файл Frame.h:

```
#ifndef FRAME
#define FRAME

#include <string>

class Frame
{
public:
    Frame();
    Frame(std::string data, Frame* linkedFrame);

    std::string getData();
    Frame* getLinkedFrame();

    void setData(std::string data);
    void setLinkedFrame(Frame* linkedFrame);

private:
    std::string data;
    Frame* linkedFrame;
};

#endif // !FRAME
```

Файл Frame.cpp:

```
#include <string>
#include "Frame.h"

Frame::Frame() : data{ "" }, linkedFrame{ nullptr } {}

Frame::Frame(std::string data, Frame* linkedFrame) : data{ data },
linkedFrame{ linkedFrame } {}

std::string Frame::getData() { return data; }

Frame* Frame::getLinkedFrame() { return linkedFrame; }
```

```
void Frame::setData(std::string data) { this->data = data; }

void Frame::setLinkedFrame(Frame* linkedFrame) { this->linkedFrame =
linkedFrame; }
```

Файл Queue.h:

```
#ifndef QUEUE
#define QUEUE

#include "Frame.h"

class Queue
{
  public:
    Queue();
    ~Queue();

    void pushFrame();
    Frame *popFrame();
    Frame *peekFrame();
    void display();
    bool isEmpty();
    void deleteFrame();
    void addAfter();

  private:
    Frame *firstFrame;
    Frame *lastFrame;
};

#endif // !QUEUE
```

Файл Queue.cpp:

```
#include <iostream>
#include <string>
#include "Queue.h"

Queue::Queue() : firstFrame{ nullptr }, lastFrame{ nullptr } {}

Queue::~Queue()
{
    if (firstFrame == nullptr) { return; }
    Frame* currFrame = firstFrame;
    while (currFrame->getLinkedFrame() != nullptr) {
        Frame* nextFrame = currFrame->getLinkedFrame();
        delete currFrame;
        currFrame = nextFrame;
    }
    delete currFrame;
}

void Queue::pushFrame()
{
    std::cout << "Enter frame data: ";
    std::string data;
    std::cin >> data;
    Frame* newFrame = new Frame(data, nullptr);
    if (firstFrame == nullptr) {
```

```
        firstFrame = newFrame;
    }
    else {
        lastFrame->setLinkedFrame(newFrame);
    }
    lastFrame = newFrame;
}

void Queue::addAfter() {
  if (lastFrame == nullptr) {
    std::cout << "Error: accessing empty stack." << std::endl;
    return;
  }

  std::string afterData;
  std::cout << "Enter frame data: ";
  std::cin >> afterData;

  Frame* prev = nullptr;
  Frame* curr = lastFrame;
  while (curr != nullptr) {
    if (curr->getData() == afterData) {

      std::cout << "Enter data after " << afterData << ": ";
      std::string newData;
      std::cin >> newData;

      Frame* newFrame = new Frame(newData, curr->getLinkedFrame());
      curr->setLinkedFrame(newFrame);
      return;
    }
    prev = curr;
    curr = curr->getLinkedFrame();
  }
  std::cout << "Error: element not found." << std::endl;
}

Frame* Queue::popFrame()
{
    if (firstFrame == nullptr) {
        std::cout << "Error: accessing empty queue." << std::endl;
        return nullptr;
    }
    Frame* returnFrame = firstFrame;
    firstFrame = firstFrame->getLinkedFrame();
    return returnFrame;
}

Frame* Queue::peekFrame()
{
    if (firstFrame == nullptr) { std::cout << "Error: accessing empty queue."
<< std::endl; }
    return firstFrame;
}

void Queue::display()
{
    if (firstFrame == nullptr) {
        std::cout << "Error: accessing empty queue." << std::endl;
        return;
    }
    Frame* currFrame = firstFrame;
    int index = 0;
    while (currFrame->getLinkedFrame() != nullptr) {
```

```
        std::cout << "Frame " << index++ << ": " << currFrame->getData() <<
std::endl;
        currFrame = currFrame->getLinkedFrame();
    }
    std::cout << "Frame " << index << ": " << currFrame->getData() <<
std::endl;
}

void Queue::deleteFrame()
{
    if (lastFrame == nullptr)
    {
        std::cout << "Error: accessing empty stack." << std::endl;
        return;
    }
    std::cout << "Enter frame data: ";
    std::string data;
    std::cin >> data;

    Frame *prev = nullptr;
    Frame *curr = lastFrame;
    while (curr != nullptr)
    {
        if (curr->getData() == data)
        {
            if (prev == nullptr)
            {
                lastFrame = curr->getLinkedFrame();
            }
            else
            {
                prev->setLinkedFrame(curr->getLinkedFrame());
            }
            delete curr;
            return;
        }
        prev = curr;
        curr = curr->getLinkedFrame();
    }
    std::cout << "Error: element not found." << std::endl;
}

bool Queue::isEmpty()
{
    if (firstFrame == nullptr) { return true; }
    else { return false; }
}
```

Файл Stack.h:

```
#ifndef STACK
#define STACK

#include "Frame.h"

class Stack
{
  public:
    Stack();
    ~Stack();

    void pushFrame();
```

```
    Frame *popFrame();
    Frame *peekFrame();
    void display();
    bool isEmpty();
    void deleteFrame();
    void addAfter();

  private:
    Frame *lastFrame;
};

#endif // !STACK
```

Файл Stack.cpp:

```
#include "Stack.h"
#include <iostream>
#include <string>

Stack::Stack() : lastFrame{nullptr}
{
}

Stack::~Stack()
{
    if (lastFrame == nullptr)
    {
        return;
    }
    Frame *currFrame = lastFrame;
    while (currFrame->getLinkedFrame() != nullptr)
    {
        Frame *nextFrame = currFrame->getLinkedFrame();
        delete currFrame;
        currFrame = nextFrame;
    }
    delete currFrame;
}

void Stack::pushFrame()
{
    std::cout << "Enter frame data: ";
    std::string data;
    std::cin >> data;
    Frame *newFrame = new Frame(data, lastFrame);
    lastFrame = newFrame;
}

void Stack::addAfter() {
  if (lastFrame == nullptr) {
    std::cout << "Error: accessing empty stack." << std::endl;
    return;
  }

  std::string afterData;
  std::cout << "Enter frame data: ";
  std::cin >> afterData;

  Frame* prev = nullptr;
  Frame* curr = lastFrame;
  while (curr != nullptr) {
    if (curr->getData() == afterData) {
```

```
      std::cout << "Enter data after " << afterData << ": ";
      std::string newData;
      std::cin >> newData;

      Frame* newFrame = new Frame(newData, curr->getLinkedFrame());
      curr->setLinkedFrame(newFrame);
      return;
    }
    prev = curr;
    curr = curr->getLinkedFrame();
  }
  std::cout << "Error: element not found." << std::endl;
}

Frame *Stack::popFrame()
{
    if (lastFrame == nullptr)
    {
        std::cout << "Error: accessing empty stack." << std::endl;
        return nullptr;
    }
    Frame *returnFrame = lastFrame;
    lastFrame = lastFrame->getLinkedFrame();
    return returnFrame;
}

Frame *Stack::peekFrame()
{
    if (lastFrame == nullptr)
    {
        std::cout << "Error: accessing empty stack." << std::endl;
    }
    return lastFrame;
}

void Stack::display()
{
    if (lastFrame == nullptr)
    {
        std::cout << "Error: accessing empty stack." << std::endl;
        return;
    }
    Frame *currFrame = lastFrame;
    int index = 0;
    while (currFrame->getLinkedFrame() != nullptr)
    {
        std::cout << "Frame " << index++ << ": " << currFrame->getData() <<
std::endl;
        currFrame = currFrame->getLinkedFrame();
    }
    std::cout << "Frame " << index << ": " << currFrame->getData() <<
std::endl;
}

void Stack::deleteFrame()
{
    if (lastFrame == nullptr)
    {
        std::cout << "Error: accessing empty stack." << std::endl;
        return;
    }
    std::cout << "Enter frame data: ";
    std::string data;
```

```
    std::cin >> data;

    Frame *prev = nullptr;
    Frame *curr = lastFrame;
    while (curr != nullptr)
    {
        if (curr->getData() == data)
        {
            if (prev == nullptr)
            {
                lastFrame = curr->getLinkedFrame();
            }
            else
            {
                prev->setLinkedFrame(curr->getLinkedFrame());
            }
            delete curr;
            return;
        }
        prev = curr;
        curr = curr->getLinkedFrame();
    }
    std::cout << "Error: element not found." << std::endl;
}

bool Stack::isEmpty()
{
    if (lastFrame == nullptr)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Файл Unidirectional_Lists.cpp:

```
#include <iostream>
#include "Stack.h"
#include "Queue.h"

using namespace std;

int main()
{
    Stack stack = Stack();
    stack.pushFrame();
    stack.pushFrame();
    std::cout << stack.popFrame() << std::endl;
    stack.pushFrame();
    stack.deleteFrame();
    stack.addAfter();
    cout << "Stack:" << endl;
    stack.display();
    Queue queue = Queue();
    queue.pushFrame();
    queue.pushFrame();
    std::cout << queue.popFrame() << std::endl;
    queue.pushFrame();
    queue.deleteFrame();
```

```
    queue.addAfter();
    queue.pushFrame();
    cout << "Queue:" << endl;
    queue.display();
}
```

Вывод: в результате работы мы познакомились с односвязными списками, стеками и очередями, реализовали их используя ЯП C++.

Основная литература

1. Зыков, С. В. Введение в теорию программирования. Объектно-ориентированный подход : учебное пособие / С. В. Зыков. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 187 с. — ISBN 978-5-4497-0926-4. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/102007.html.
2. Павловская, Т. А. Программирование на языке высокого уровня C# : учебное пособие / Т. А. Павловская. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 245 с. — Текст : электронный — URL: http://www.iprbookshop.ru/102051.html.
3. Биллиг, В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008) : учебник / В. А. Биллиг. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 409 с. — Текст : электронный — URL: http://www.iprbookshop.ru/102029.html.
4. Горелов, С. В. Современные технологии программирования: разработка Windows-приложений на языке C#. В 2 томах. Т.I : учебник / С. В. Горелов ; под редакцией П. Б. Лукьянова. — Москва : Прометей, 2019. — 362 с. — Текст : электронный — URL: http://www.iprbookshop.ru/94532.html.
5. Горелов, С. В. Современные технологии программирования: разработка Windows-приложений на языке C#. В 2 томах. Т.II : учебник / С. В. Горелов ; под редакцией П. Б. Лукьянова. — Москва : Прометей, 2019. — 378 с. — Текст : электронный — URL: http://www.iprbookshop.ru/94533.html.