# Лабораторная работа 2. Наследование классов в С++

## Задание

Создать базовый класс CList (линейный однонаправленный cписок) с полями: указатели на начало списка, на его конец и на следующий элемент; информационная часть – целое число. В производных классах – CQueue (очередь) и CStack (стек) – должны быть определены методы вставки и удаления узла в соответствии с дисциплиной обслуживания соответствующего класса.

## Диаграмма классов

C\_List

CQueue

CStack

Client

Container

## Перечень сигнатур методов с комментариями согласно заданию

**C\_List.h**

#include "Container.h"

#pragma once

class C\_List

{

protected:

Container\* head;

Container\* tail;

public:

virtual void push(int) = 0;//создание виртуального метода для вставки в список

virtual int pop() = 0; //создание виртуального метода для удаления элемента из списока

virtual bool empty() = 0;//проверка на пустоту списка

C\_List(void);

~C\_List(void);

};

## Листинг

//Class Container

**Container.h**

#include "Container.h"

Container::Container(void)

{

next = nullptr;

}

Container::Container(int value\_, Container\* next\_)

{

value = value\_;

next = next\_;

}

Container::~Container(void)

{

}

**Container.cpp**

#include "Container.h"

Container::Container(void)

{

next = nullptr;

}

Container::Container(int value\_, Container\* next\_)

{

value = value\_;

next = next\_;

}

Container::~Container(void)

{

}

//Class C\_List

**C\_List.h**

#include "Container.h"

#pragma once

class C\_List

{

protected:

Container\* head;

Container\* tail;

public:

virtual void push(int) = 0;

virtual int pop() = 0;

virtual bool empty() = 0;

C\_List(void);

~C\_List(void);

};

**C\_List.cpp**

#include "C\_List.h"

C\_List::C\_List(void)

{

head = nullptr;

tail = nullptr;

}

C\_List::~C\_List(void)

{

}

//Class Queue

**Queue.h**

#pragma once

#include "c\_list.h"

class CQueue :

public C\_List

{

public:

CQueue(void);

~CQueue(void);

void push(int value);

int pop();

bool empty();

};

**Queue.cpp**

#include "Queue.h"

CQueue::CQueue(void)

{

}

CQueue::~CQueue(void)

{

}

void CQueue::push(int value\_)

{

if (head != nullptr)

{

Container\* tmp = new Container(value\_, nullptr);

tail->next = tmp;

tail = tmp;

}

else

{

head = new Container(value\_, nullptr);

tail = head;

}

}

int CQueue::pop()

{

int response;

if (head != nullptr)

{

response = head->value;

Container\* tmp = head;

head = head->next;

delete tmp;

}

return response;

}

bool CQueue::empty()

{

if (head == nullptr)

{

return true;

}

else

{

return false;

}

}

//Class Stack

**Stack.h**

#pragma once

#include "c\_list.h"

class CStack :

public C\_List

{

public:

void push(int item);

int pop();

bool empty();

CStack(void);

~CStack(void);

};

**Stack.cpp**

#include "Stack.h"

CStack::CStack(void){ }

CStack::~CStack(void){ }

void CStack::push(int value\_)

{

if (head != nullptr)

{

Container\* tmp = new Container(value\_, tail);

tail = tmp;

}

else

{

head = new Container(value\_, nullptr);

tail = head;

}

}

int CStack::pop()

{

int response;

if (tail != nullptr)

{

response = tail->value;

Container\* tmp = tail;

tail = tail->next;

delete tmp;

}

return response;

}

bool CStack::empty()

{

if (tail == nullptr)

{

return true;

}

else

{

return false;

}

}

**//main:**

#include <iostream>

#include "Stack.h"

#include "Queue.h"

using namespace std;

int main()

{

CQueue queue;

CStack stack;

for (int i = 0; i < 10; i++)

{

queue.push(i);

stack.push(i);

}

cout << "Work of the Queue:" << endl;

for (int i = 0; i < 20; i++)

{

if (!queue.empty())

{

cout << queue.pop() << ' ';

}

}

cout << endl;

cout << "Work of the Stack:" << endl;

for (int i = 0; i < 20; i++)

{

if (!stack.empty())

{

cout << stack.pop() << ' ';

}

}

return 0;

}

**Вывод на консоль:**

Work of the Queue:

0 1 2 3 4 5 6 7 8 9

Work of the Stack:

9 8 7 6 5 4 3 2 1 0

Знание кода Жиденко Н.Н. Отметки

Контрольные вопросы Жиденко Н.Н. Отметки

Автор Преподаватель

Жиденко Н.Н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись) Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись \_\_\_\_\_\_\_\_ Дата