МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Списки. Стеки. Очереди.

# Отчет

по лабораторной работе 1



по дисциплине

Алгоритмы и структуры данных

(наименование дисциплины)

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бубнов Е.А

(подпись) (фамилия, и.,о.)

19-ИВТ-2

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020

## Задание

Задача 9.

N серых и M белых мышей сидят по кругу. Кошка ходит по кругу по часовой стрелке и съедает каждую S -тую мышку. В первый раз счет начинается с серой мышки. Составить алгоритм определяющий порядок в котором сидели мышки, если через некоторое время осталось K серых и L белых мышей.

Определить:

-Определить номер первой серой мыши

-Определить порядок съедения серых и белых мышек

## Листинг

Файл Исходный код.cpp

#include <iostream>

#include <cstdlib>

#define MAXLEN\_BUFFER 255

#define STOP 's'

using std::cout;

using std::cin;

using std::endl;

//список

template<typename LT>

class List

{

private:

//ячейка

template<typename NT>

class Node

{

public:

NT data; //данные

Node\* next; //указатель на следующий элемент

Node(NT \_data=NT(), Node\* \_next=NULL) : data(\_data), next(\_next) {}

~Node(){}

};

Node<LT>\* head; //указатель на говолу

Node<LT>\* end; //указатель на конец

int SIZE; //размер списка

public:

List() : head(NULL), end(NULL), SIZE(0) {}

~List();

void push\_back(const LT& \_data); // добавление в конец списка

void pop (int index); // удаление index элемнта

void print (); //вывод всего списка

LT get (const int index); // получение значения index элемента

bool empty() { return (SIZE == 0)? true : false; } // проверка на пустоту списка

int size() { return SIZE; } // определение размера списка

};

void create(List<bool>& live\_mice, int\* n\_white\_mouse, int\* n\_grey\_mouse, int\* first\_mouse);

void find\_first\_mouse(bool\* have\_first\_mouse, int\* first\_mouse, int n);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*начало главной части\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()

{

List<bool> live\_mice; //живые мышки

List<bool> dead\_mice; //съеденный мышки

int n\_white\_mouse; // кол-во белых мышек

int n\_grey\_mouse; // кол-во серых мышек

int first\_mouse; //номер серой мыши

//создание списка мышек

create(live\_mice, &n\_white\_mouse, &n\_grey\_mouse, &first\_mouse);

//вывод результата создания списка

cout << "Live mice: ";

live\_mice.print(); cout << endl;

cout << "-white = " << n\_white\_mouse << endl

<< "-grey = " << n\_grey\_mouse << endl;

//начало обеда

int cat = first\_mouse; //кот

//кот ходит по кругу

while (!live\_mice.empty())

{

//добавляю очередную мышь в список съеденных

dead\_mice.push\_back(live\_mice.get(cat));

live\_mice.pop(cat);

//если кот не съел мышей в начале

if (cat+1 > live\_mice.size())

cat = 0; // перходи в начало

}

cout << "Dead mice: ";

dead\_mice.print(); cout << endl;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*конец главной части\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

template<typename LT>

List<LT>::~List()

/\*Удаляем список\*/

{

Node<LT>\* tmp;

while (head != NULL)

{

tmp = head;

head = head->next;

delete tmp;

}

}

template<typename LT>

void List<LT>::print()

/\*выводим список\*/

{

Node<LT>\* tmp = head; //создаем временнный объект для обхода

//пока не конец списка выводи элемнты

while (tmp != NULL)

{

cout << tmp->data << ' ';

tmp = tmp->next;

}

}

template<typename LT>

void List<LT>::push\_back(const LT& \_data)

/\*Добавление в конец списка\*/

{

//если элемнтов нет

if (SIZE == 0)

{

head = new Node<LT>(); //создаем елемент

head->data = \_data; //заполняем

end = head; //присваем новый хвост

}

else

{

Node<LT>\* tmp = new Node<LT>(); //создаем елемент

tmp->data = \_data; //заполняем

end->next = tmp; //говорим что n-ый элемент указывает на наш новый элемент

end = tmp; //присваем новый хвост

}

SIZE++; //увеливаем количтво элементов

}

template<typename LT>

void List<LT>::pop(int index)

/\* Удаление index элемента \*/

{

if ((head != NULL) && (index >= 0) && (index < SIZE)) // если по этому номеру что-то лежит и этот элемент внутри списка

{

// Mass - объекты, которые хранятся в списке

Node<LT> \*tmp = head, \*helping = head;

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

{

helping = tmp; // предыдущее значение temp

tmp = tmp->next;

}

if (tmp == head) // если элемент который надо удалить первый

{

head = tmp->next;

}

else

{

helping->next = tmp->next;

}

delete(tmp);

SIZE--; // уменьшаем размер списка

}

else

{

throw -1;

}

}

template<typename LT>

LT List<LT>::get(const int index)

{

//если индекс элемента правильный

if (index >= 0 && index < SIZE)

{

//создаю временный указатель на элемент для обхода списка

Node<LT>\* tmp = head;

int count = 0; //счетчик

while (count < index) //пока не найден нужный элемент

{

tmp = tmp->next; // переходи на следующий элемент

count++;

}

return tmp->data; //возвращаю найденный данные

}

//индекс введен с ошибкой, возврат исключения

throw -1;

}

void create(List<bool>& live\_mice, int\* n\_white\_mouse, int\* n\_grey\_mouse, int\* first\_mouse)

/\* Создает список мышей \*/

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*n\_white\_mouse = \*first\_mouse = \*first\_mouse = 0;

char data; //переменная для считывания цветы мыши

bool have\_first\_mouse = false; //сигнал что первая серая мышка нашлась

//пока нет ошибок или конца

int i = 0;

while (true)

{

//ввод данных

cout << "Input color mouse(0 - white, 1 - grey): ";

cin >> data;

//если поток пуст и значение 0 или 1

if ((data == '1' || data == '0') && cin.peek() == '\n')

{

//Добавляй в список

live\_mice.push\_back(bool(data-'0'));

//если 1 то белая мышь

if (live\_mice.get(i) == 0)

(\*n\_white\_mouse)++;

else

{

(\*n\_grey\_mouse)++; //иначе серая

//поиск первой серой мышки

find\_first\_mouse(&have\_first\_mouse, first\_mouse, i);

}

i++;

}

else if (data == STOP && cin.peek() == '\n') //заканчиваем ввод

{

break;

}

else

{

//очищай буфер

cin.ignore(MAXLEN\_BUFFER, '\n');

cin.clear();

//выводи ошибку

cout << "ERROR!" << endl;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

void find\_first\_mouse(bool\* have\_first\_mouse, int\* first\_mouse, int n)

/\*поиск первой серой мышки\*/

{

//если мышка еще не найдена, то запиши

if (!(\*have\_first\_mouse))

{

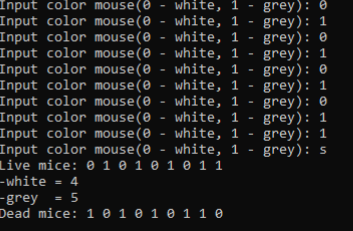
\*first\_mouse = n;

\*have\_first\_mouse = true;

}

}

## Скриншоты



Обработка ошибок:

