**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Кафедра безопасности информационных систем**

**ОТЧЁТ**

по практической работе № 7 на тему:   
**«Разработка и применение циклического списка.**

**Задача Иосифа.»**

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы ИСТ-312, Кандиков М.В.

«14» Октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_/М.В. Кандиков /

Принял: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«15» Октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_/ И.А. Моисеев /

**Содержание**

**1. Титульный лист 1**

**2. Содержание 2**

**3. Основная часть 3**

3.1. Цель работы 3

3.2. Теоретическая часть 3

3.3. Практическая часть 4

3.4. Результаты выполнения программы 6

3.5. Выводы 7

**4. Приложение 9**

**Основная часть**

**Цель работы:**

Целью данной практической работы является изучение и применение структуры данных «циклический список» для решения классической задачи Иосифа. Мы должны разработать алгоритм, который на базе односвязного кольцевого списка позволяет моделировать процесс последовательного удаления элементов списка и нахождения последнего оставшегося элемента. Важно освоить принципы работы с циклическими списками, понять их особенности и научиться реализовывать основные операции, такие как вставка, удаление и обход элементов списка.

**Теоретическая часть:**

Циклический список — это разновидность линейного списка, в котором последний элемент списка указывает на первый, образуя замкнутую структуру. В односвязном кольцевом списке каждый элемент хранит указатель только на следующий элемент, а последний элемент замыкается на первый. Циклические списки широко применяются в задачах, связанных с круговыми процессами, таких как организация буферов данных, реализация очередей и решение задач, подобных задаче Иосифа.

Задача Иосифа является классической задачей в информатике и моделирует круг людей, где через определённые интервалы один человек исключается из круга. Процесс продолжается до тех пор, пока не останется один человек. Задачу можно эффективно решать с помощью циклического списка, который позволяет обходить элементы по кругу и удалять их по мере необходимости.

**Практическая часть:**

Программа реализует алгоритм решения задачи Иосифа с использованием односвязного кольцевого списка. Вначале создаётся круг из 20 участников, где каждый участник имеет уникальный номер от 1 до 20. Для каждого значения параметра m от 2 до 20, программа моделирует процесс удаления участников с определённым шагом и выводит номер последнего оставшегося участника.

**Основные этапы работы программы:**

* **Инициализация списка**: В функции Init\_Circle() создаётся односвязный циклический список из 20 узлов. Каждый узел хранит номер участника, а последний узел указывает на первый, формируя кольцевую структуру.
* **Удаление узлов**: В функции Josephus\_Circle() происходит удаление узлов по правилу задачи Иосифа — каждый m-й участник удаляется из круга до тех пор, пока не останется один.
* **Вывод результатов**: Для каждого значения m от 2 до 20 программа выводит на экран номер последнего оставшегося участника.

Ниже приведены результаты выполнения программы:

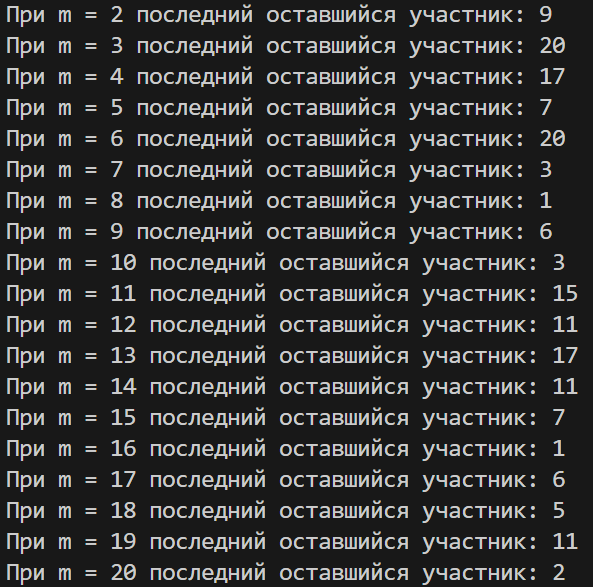


Рисунок 1. Результаты выполнения программы.

Для каждого значения m программа корректно находит последнего оставшегося участника.

**Выводы:**

Циклические списки оказались удобным и эффективным инструментом для решения задач, связанных с круговым просмотром элементов, таких как задача Иосифа. Благодаря замкнутой структуре списка, процесс удаления участников происходил последовательно и без необходимости пересоздавать или изменять структуру данных. Это делает алгоритм не только простым, но и производительным, так как удаление и обработка узлов выполняются быстро.

Работа над задачей помогла лучше понять, как важно правильно организовать данные и работать с указателями. Это позволило решить задачу с минимальными затратами ресурсов и показало, что структурированные данные значительно упрощают выполнение сложных операций. Применение циклических списков на практике продемонстрировало их значимость в различных областях, таких как управление памятью, сетевые алгоритмы и игровые приложения.

Опыт работы с циклическими структурами данных и алгоритмами для задачи Иосифа подтвердил, что такие подходы могут быть полезны для моделирования процессов с круговыми зависимостями и эффективного управления ресурсами.

**Приложение**

Листинг программы:

#include <iostream>

using namespace std;

// Структура для представления узла

struct Node {

    int data;    // значение узла (номер участника)

    Node \*next;  // указатель на следующий узел

};

// Функция для инициализации кольцевого списка

Node\* Init\_Circle(int n) {

    Node \*head = new Node; // создаем первый узел

    head->data = 1;        // присваиваем значение первому участнику

    Node \*prev = head;

    // Создаем узлы для остальных участников

    for (int i = 2; i <= n; i++) {

        Node \*new\_node = new Node;

        new\_node->data = i;

        prev->next = new\_node;

        prev = new\_node;

    }

    prev->next = head; // замыкаем список на первый узел

    return head;

}

// Функция для удаления узла с шагом m и нахождения последнего участника

int Josephus\_Circle(Node \*head, int m) {

    Node \*current = head;

    Node \*prev = nullptr;

    // Процесс удаления узлов, пока не останется один

    while (current->next != current) {

        // Пробегаем m-1 шагов

        for (int i = 1; i < m; i++) {

            prev = current;

            current = current->next;

        }

        // Удаляем текущий узел

        prev->next = current->next;

        delete current;

        current = prev->next;

    }

    int last\_person = current->data;

    delete current; // удаляем последний узел

    return last\_person;

}

int main() {

    int n = 20;  // количество участников

    // Для каждого m от 2 до 20 выводим результат

    for (int m = 2; m <= 20; m++) {

        Node \*circle = Init\_Circle(n);  // создаем новый список для каждого m

        int last\_person = Josephus\_Circle(circle, m);

        cout << "При m = " << m << " последний оставшийся участник: " << last\_person << endl;

    }

    return 0;

}