**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, Серафимович Г.П

«14» Октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_/ Г.П Серафимович /

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:**

Цель этой практической работы — изучение и использование структуры данных «циклический список» для решения классической задачи Иосифа. Нам необходимо разработать алгоритм, который на основе односвязного кольцевого списка будет моделировать процесс последовательного удаления элементов и определять последний оставшийся элемент.

**Теоретическая часть:**

Циклический список представляет собой особый вид линейного списка, где последний элемент ссылается на первый, образуя замкнутую структуру. В односвязном кольцевом списке каждый элемент содержит указатель только на следующий, а последний элемент указывает на первый. Циклические списки находят широкое применение в задачах, связанных с круговыми процессами, таких как организация буферов данных, реализация очередей и решение задач, подобных задаче Иосифа.

**Практическая часть:**

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

Основные этапы работы программы:

• Инициализация списка: В функции circle() создаётся односвязный циклический список из 20 узлов. Каждый узел содержит номер участника, а последний узел ссылается на первый, формируя кольцевую структуру.

• Удаление узлов: В функции josephus() происходит удаление узлов по правилам задачи Иосифа — каждый m-й участник исключается из круга до тех пор, пока не останется только один.

• Вывод результатов: для каждого значения m от 2 до 20 программа отображает номер последнего оставшегося участника.

Результаты выполнения программы:

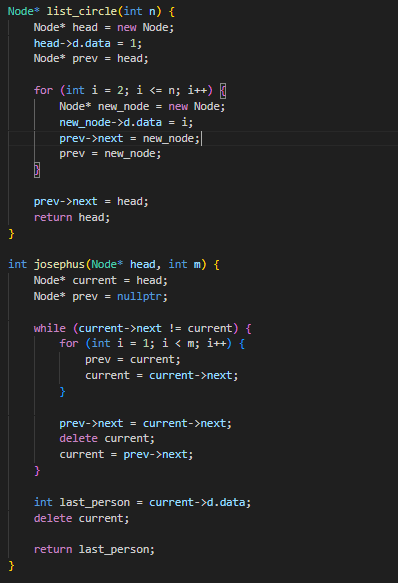


Рисунок 1.2 list\_circle and josephus

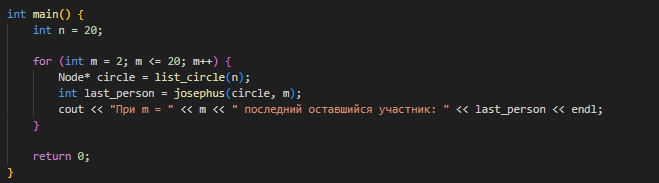


Рисунок 1.3 main

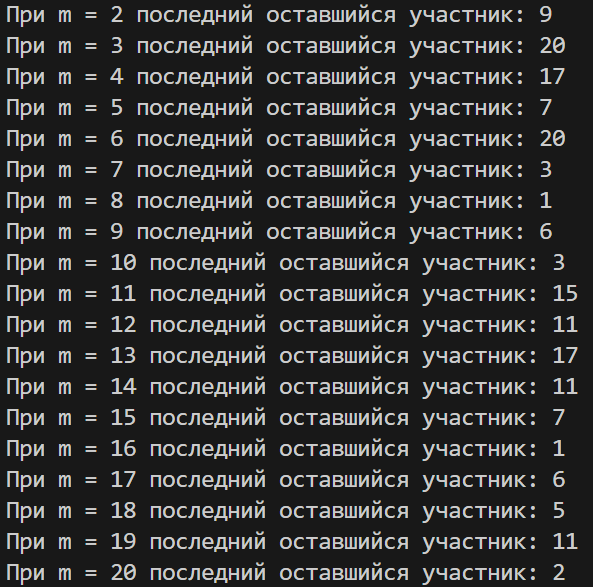


Рисунок 1.3 Результаты выполнения программы.

Для каждого значения m программа корректно находит последнего оставшегося участника.

**Выводы:**

Циклические списки оказались удобным и эффективным инструментом для решения задач, связанных с круговым обходом элементов, таких как задача Иосифа. Благодаря замкнутой структуре списка процесс удаления участников происходил последовательно, без необходимости пересоздания или изменения структуры данных. Это делает алгоритм не только простым, но и производительным, так как операции удаления и обработки узлов выполняются быстро.

**Приложение**

Листинг программы:

#include <iostream>

using namespace std;

struct Data {

int data;

};

struct Node {

Data d;

Node\* next;

};

Node\* list\_circle(int n) {

Node\* head = new Node;

head->d.data = 1;

Node\* prev = head;

for (int i = 2; i <= n; i++) {

Node\* new\_node = new Node;

new\_node->d.data = i;

prev->next = new\_node;

prev = new\_node;

}

prev->next = head;

return head;

}

int josephus(Node\* head, int m) {

Node\* current = head;

Node\* prev = nullptr;

while (current->next != current) {

for (int i = 1; i < m; i++) {

prev = current;

current = current->next;

}

prev->next = current->next;

delete current;

current = prev->next;

}

int last\_person = current->d.data;

delete current;

return last\_person;

}

int main() {

int n = 20;

for (int m = 2; m <= 20; m++) {

Node\* circle = list\_circle(n);

int last\_person = josephus(circle, m);

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

}

return 0;

}