Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Перегрузка побитовых логических операции

Задан элемент, состоящий из ячейки памяти данных **объемом один байт** и шаблона активных битов (размер также равен 1 байту). Между данными из ячеек памяти двух элементов можно выполнить побитовые логические операции умножения и сложения. От каждого элемента в операциях участвуют только те биты данных, которые соответствуют шаблону активных битов элемента.

Работа с элементами выполняется следующим образом. Первоначально создаём элементы, определяем для них содержимое ячейки памяти и значение шаблона в шестнадцатеричной системе счисления. Далее описываем логические выражения, включающие эти элементы.

Написать программу, которая моделирует работу с элементами.

В основной программе реализовать алгоритм:

1. Ввод количества элементов n.

2. В цикле для каждого элемента вводится исходное значение ячейки памяти и значение шаблона активных битов. Далее создается объект, в конструктор которого передаются значения памяти и шаблона. Каждому объекту присваивается свой номер от 1 до n.

3. В цикле, последовательно и построчно, вводится «номер первого объекта» «символ логической операции & или |» «номер второго объекта»

4. После каждого нового ввода логического выражения выполняется логическая операция, результат записывается в ячейку памяти первого элемента (объекта).

5. Цикл завершается в тот момент, когда на ввод больше нет данных.

6. Выводится результат последней операции в шестнадцатеричном формате.

Количество элементов больше или равно 2.

Использовать перегрузку логических побитовых операций, реализовав в составе описания класса.

Пояснения.

Значения в пояснении заданы в шестнадцатеричной системе счисления.

Значение логической единицы (1) в шаблоне задаёт активный бит значения из ячейки памяти. Если значение шаблона равно 15, то активными будут считаться 4-й, 2-й и 0-й биты значения из ячейки памяти.

В логической операции между двумя элементами участвуют только те активные биты ячеек памяти, позиции которых совпадают у обоих элементов (находятся на пересечении). Например, если значение шаблона одного элемента равно 0F, а другого 0C, то в логической операции участвуют только 3-й и 2-й биты обоих значений. Соответственно, при записи результата в первый элемент изменениям подвергаются только те биты, которые участвовали в операции.

Первый элемент e1: значение памяти 8F, значение шаблона 0F.

Второй элемент e2: значение памяти 02, значение шаблона 01.

Операция e1 & e2. Значение первого элемента равно 8E,

Первый элемент e1: значение памяти 8F, значение шаблона 0F.

Второй элемент e2: значение памяти 02, значение шаблона F0.

Операция e1 & e2. Значение первого элемента равно 8F,

1.1 Описание входных данных

Первая строка содержит значение количества элементов n:

«Натуральное значение»

Далее n строк содержат

«Шестнадцатеричное значение» «Шестнадцатеричное значение»

Начиная с n + 2 строки:

«Натуральное значение» «Знак операции» «Натуральное значение»

1.2 Описание выходных данных

«Шестнадцатеричное значение»

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- функция main для .

3 ОПИСАНИЕ АЛГОРИТМОВ

Согласно этапам разработки, после определения необходимого инструментария в разделе «Метод», составляются подробные описания алгоритмов для методов классов и функций.

3.1 Алгоритм функции main

Функционал: Основная функция.

Параметры: нет.

Возвращаемое значение: Целочисленное.

Алгоритм функции представлен в таблице 1.

Таблица 1 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Ввод количества элементов n	2
2		В цикле для каждого элемента вводится исходное значение ячейки памяти и значение шаблона активных битов. Далее создается объект, в конструктор которого передаются значения памяти и шаблона. Каждому объекту присваивается свой номер от 1 до n	3
3		В цикле, последовательно и построчно, вводится «номер первого объекта» «символ логической операции & или |» «номер второго объекта»	4
4		После каждого нового ввода логического выражения выполняется логическая операция, результат записывается в ячейку памяти первого элемента (объекта).	5
5		Цикл завершается в тот момент, когда на ввод больше нет данных.	6
6		Выводится результат последней операции в шестнадцатеричном формате.	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-1.

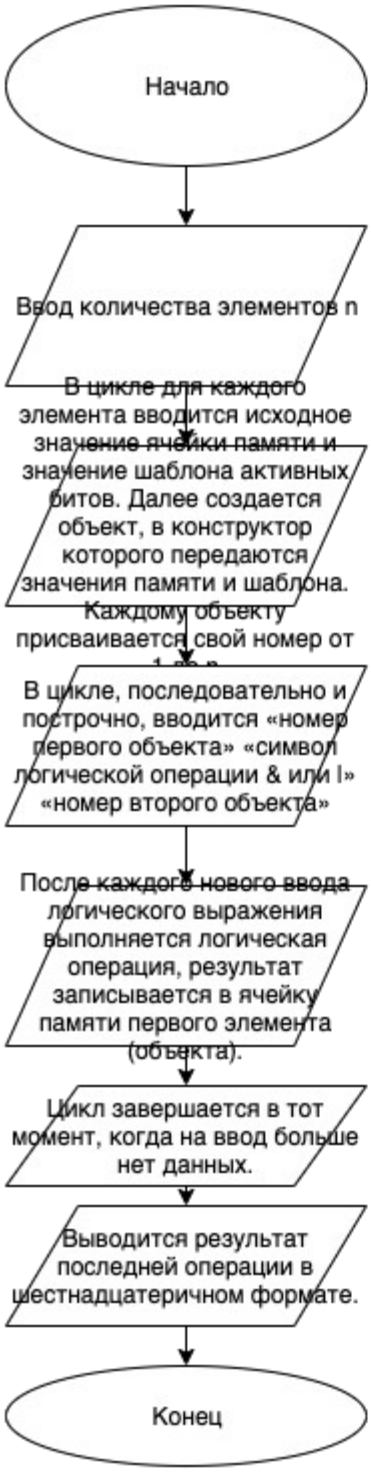

Рисунок 1 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

```
#include <iostream>
#include <vector>

struct element
{
   unsigned char data;
   unsigned char data_template;

   element& operator|(const element& object)
   {
      int origin, destination;
      for(int i = 0; i< 8; i++)
      {
         origin = (object.data_template >> i) & 1;
         destination = (data_template >> i) & 1;

         if(origin & destination)
         {
            origin = (object.data >> i) & 1;
            destination = (data >> i) & 1;
            if(!destination) data |= (origin << i);
         }
      }
      return *this;
   }

   element& operator|=(const element& object)
   {
      return operator|(object);
   }

   element& operator&=(const element& object)
   {
      int origin, destination;
      for(int i = 0; i< 8; i++)
      {
         origin = (object.data_template >> i) & 1;
         destination = (data_template >> i) & 1;
```

```
            if(origin & destination)
            {
               origin = (object.data >> i) & 1;
               destination = (data >> i) & 1;
               if(origin & destination) data |= (1 << i);
               else data &= ~(1 << i);
            }
         }
         return *this;
      }

      element(unsigned char _data, unsigned char _data_template)
      {
         data = _data;
         data_template = _data_template;
      }

      element operator|(const element& object) const
      {
         unsigned char new_data = data | object.data;
         unsigned char new_data_template = data_template | object.data_template;
         return element(new_data, new_data_template);
      }
   };

   unsigned char in_bits(std::string value)
   {
      std::string alphabet = "0123456789ABCDEF";
      return alphabet.find(value.at(1)) + 16 * alphabet.find(value.at(0));
   }

   std::string from_bits(unsigned char value)
   {
      std::string alphabet = "0123456789ABCDEF";
      return   std::string(1,   alphabet.at(value   >>   4))   +   std::string(1,
   alphabet.at(value - 16 * (value >> 4)));
   }
   int main()
   {
      char sign;
      int head, sub, amount = 0;
      std::string data[2];
      std::vector<element> elements;

      while(amount < 2 || abs(amount) != amount) std::cin >> amount;
      for(; amount > 0; amount--)
      {
         std::cin >> data[0] >> data[1];
         elements.push_back(element(in_bits(data[0]), in_bits(data[1])));
      }

      while(std::cin >> head)
      {
         std::cin >> sign >> sub;
```

```
            if(sign == '&') elements[head - 1] &= elements[sub - 1];
            else if(sign == '|') elements[head -  1] |= elements[sub - 1];
        }
        std::cout << from_bits(elements.at(head - 1).data);
        return(0);
    }
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 2.

Таблица 2 – Результат тестирования программы

Входные данные	**Ожидаемые выходные данные**	**Фактические выходные данные**
2 1A 2B 3C 4D 1 | 2 1 & 2	1A	1A

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19 Единая система программной документации.

2. Методическое пособие студента для выполнения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: https://mirea.aco-avrora.ru/student/files/methodichescoe_posobie_dlya_laboratornyh_rabot_3.pdf (дата обращения 05.05.2021).

3. Приложение к методическому пособию студента по выполнению заданий в рамках курса «Объектно-ориентированное программирование» [Электронный ресурс]. URL: https://mirea.aco-avrora.ru/student/files/Prilozheniye_k_methodichke.pdf (дата обращения 05.05.2021).

4. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2019. — 624 с.

5. Видео лекции по курсу «Объектно-ориентированное программирование» [Электронный ресурс]. АСО «Аврора».

6. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие /Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).