# **2. Лабораторная работа № 2. Изучение перегрузки стандартных операций в языке Си++**

## **2.1. Цель и задачи работы, требования к результатам ее выполнения**

Цель работы состоит в овладении навыками разработки программ на языке Си++, использующих перегрузку стандартных операций. Для достижения цели необходимо выполнить следующие задачи:

- изучить необходимые учебные материалы, посвященные перегрузке стандартных операций в языке Си++ ;

- разработать программу на языке Си++ для решения заданного варианта задания;

- отладить программы;

- выполнить решение контрольного примера с помощью программы и ручной расчет контрольного примера;

- подготовить отчет по лабораторной работе.

## **2.2. Краткая характеристика объекта изучения**

Перегрузка операций в языке Си++ это возможность распространения действия стандартных операций на операнды, для которых эти операции первоначально не предназначались. Это возможно, если хотя бы один из операндов является объектом класса, для этого создается специальная, так называемая, оператор- функция, которая может быть как членом класса, так и функцией, не принадлежащей классу.

Формат определения оператор- функции имеет вид:

*<тип\_возвращаемого\_значения>*

*operator <знак\_операции>*

*(спецификация\_параметров)*

*{*

*операторы\_тела\_функции*

*}*

Существует три способа перегрузки:

* оператор-функция определяется как функция, не принадлежащая классу;
* оператор-функция определяется как функция класса;
* оператор-функция определяется как дружественная функция класса.

Особенности перегрузки операций:

* можно перегружать только стандартные операции, например, нельзя перегрузить операцию ‘*\*\*’* (возведение в степень в языке Фортран, отсутствует в Си++);
* не допускают перегрузки операции: *‘.’, ‘.\*’, ‘?:’, ‘::’, ‘sizeof’, ‘#’, ‘##’*;
* при перегрузке сохраняется арность операций (унарная операция остается унарной, а бинарная – бинарной);
* бинарная операция перегружается либо как функция, не принадлежащая классу с двумя параметрами, один обязательно объект (ссылка на объект) класса, или как функция класса с одним параметром, первым операндом операции выступает объект класса, для которого вызывается функция;
* бинарные операции *‘=‘, ‘[]’, ‘->’* должны обязательно определяться как компонентные функции класса;
* унарная операция перегружается либо как функция, не принадлежащая классу с одним параметром - объектом (ссылкой на объект) класса, или как функция класса без параметров, операндом операции выступает объект класса, для которого вызывается функция.

## **2.3. Задачи и порядок выполнения работы**

Студент в работе создает класс и необходимые оператор- функции для перегрузки заданных в своем варианте операций. Особое внимание обратить на способы перегрузки унарных и бинарных операций и параметры оператор- функций для этих операций, когда оператор функция является членом класса и когда не является. Знать те случаи, когда оператор функция должна быть обязательно членом класса и случаи, когда оператор- функция обязательно не принадлежит классу. При защите работы необходимо обосновать выбор способа определения оператор- функции – внутри класса или вне его. При необходимости студент выполняет ручной расчет для проверки работы программы для задачи небольшой размерности. Результаты работы программы, ручного расчета представляются в отчете.

## **2.4. Форма отчета по лабораторной работе**

Отчет должен содержать: титульный лист, цель работы, условие задачи, текст программы с комментариями, при необходимости ручной расчет контрольного примера и результаты решения контрольного примера программой для проверки правильности работы алгоритма, выводы по работе.

## **2.5. Вопросы для самоконтроля**

1. Способы перегрузки операций в Си++.

2. Перегрузка бинарной операции, когда первый операнд не является объектом класса.

3. Перегрузка операции с помощью дружественной функции класса.

4. Операции, для перегрузки которых оператор- функция обязательно должна принадлежать классу.

5. Перегрузка операторов инкремента и декремента в постфиксной и префиксной формах. Различия.

6. Использование ключевого слова const в методах класса.

7. Перегрузка операторов присваивания и перемещения.

**Варианты заданий для лабораторной работы № 2**

Дан класс (например, с именем *Vector*), задающий вектор размерности *n*. Поля класса: указатель на массив, задающий вектор (тип элемента *int* или *double* в зависимости от варианта), массив должен создаваться динамически, число элементов (размерность) вектора (тип *int*). Класс включает: конструктор без параметров, задающий пустой вектор (число элементов равно 0), конструктор, создающий объект вектор на основе обычного одномерного массива размерности *n*, конструктор копирования, конструктор перемещения, деструктор.

Необходимо перегрузить операции и продемонстрировать их работу. Перегрузить операцию [] (обращение к элементу вектора по индексу), операцию = (присваивание с копированием), операцию = (присваивание с перемещением), а также операцию вставки (<<) объекта в поток cout или в файл (объект класса ostream) и операцию извлечения (>>) объекта из потока cin или из файла (объект класса istream). *Также продемонстрировать разницу между конструктором копирования и конструктором перемещения и между операциями присваивания с копированием и перемещением. Исходные коды класса разместить в двух файлах: в заголовочном файле класса и файле реализации класса.*

*При выполнении работы все входные данные читаются из текстового файла input.txt (создать этот файл любым текстовым редактором), результаты выводятся в файл output.txt. В отчете представить содержимое этих файлов.*

Индивидуальные операции для перегрузки каждым студентом представлены в таблице 1.

Таблица 1 – Варианты заданий

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Описание операции перегруженной операции | Тип элемента вектора (массива) | Типы операндов и результата для перегруженной операции | | | № варианта |
| Первый операнд | Второй операнд | Результат |  |
| + сложение векторов одинаковой размерности, на выходе вектор такой же размерности элемент которого равен сумме соответствующих элементов двух векторов | double | Vector | Vector | Vector | 1 |
| Vector | double \* | Vector | 2 |
| double \* | Vector | Vector | 3 |
| + сложение векторов, на выходе вектор, длина которого сумме длин векторов вначале идут элементы первого вектора, затем второго, если один из векторов задан обычным массивом, то считать, что его длина равна длине вектора, заданным объектом класса | double | Vector | Vector | Vector | 4 |
| Vector | double \* | Vector | 5 |
| double \* | Vector | Vector | 6 |
| Постфиксная форма – (декремент), при этом каждый элемент исходного вектора уменьшается на 1, результат операции «старый» (неизмененный) вектор. | int | Vector | \_\_\_\_\_ | Vector | 7 |
| \* умножение вектора на число, на выходе вектор такой же размерности, каждый элемент которого равен произведению соответствующего элемента исходного вектора на число | double | Vector | double | Vector | 8 |
| double | Vector | Vector | 9 |
| Префиксная форма ++ (инкремент), при этом каждый элемент исходного вектора увеличивается на 1, результат операции - измененный вектор. | int | Vector | \_\_\_\_\_ | Vector | 10 |
| \* скалярное произведение векторов (одинаковой размерности), на выходе значение этого произведения | double | Vector | Vector | double | 11 |
| Vector | double \* | double | 12 |
| double \* | Vector | double | 13 |
| Постфиксная форма ++ (инкремент), при этом каждый элемент исходного вектора увеличивается на 1, результат операции «старый» (неизмененный) вектор. | int | Vector | \_\_\_\_\_ | Vector | 14 |
| ^ побитовая операция исключающая ИЛИ с двумя векторами одинаковой размерности, на выходе вектор такой же размерности элемент, которого равен битовой операции ^ соответствующих элементов двух векторов | int | Vector | Vector | Vector | 15 |
| Vector | int \* | Vector | 16 |
| int \* | Vector | Vector | 17 |
| Префиксная форма -- (декремент),, при этом каждый элемент исходного вектора уменьшается на 1, результат операции измененный вектор. | int | Vector | \_\_\_\_\_ | Vector | 18 |
| ^ логическая операция (исключающая ИЛИ) с двумя векторами одинаковой размерности, на выходе вектор такой же размерности элемент, которого равен логическая операции ^ соответствующих элементов двух векторов | bool | Vector | Vector | Vector | 19 |
| Vector | bool \* | Vector | 20 |
| bool \* | Vector | Vector | 21 |
| < сравнение двух векторов (массивов) сравнение проводится последовательно по элементам. Результат истинно, если элемент первого операнда меньше элемента второго элемента, если элемент первого операнда больше элемента второго элемента, результат ложь, в случае равенства переход к следующий паре элементов. Если в каком либо операнде элементы закончились, то результат ложь. Если один из векторов задан обычным массивом, то считать, что его длина равна длине вектора, заданным объектом класса. | int | Vector | Vector | bool | 22 |
| Vector | int \* | bool | 23 |
| int \* | Vector | bool | 24 |
| ^ логическая операция с двумя векторами произвольной размерности, на выходе вектор состоящий из элементов, входящих только в один вектор (дубликаты удаляются внутри каждого вектора, а также если элемент «а» есть и в первом и во втором векторе, то в итоговый он не включается) | int | Vector | Vector | Vector | 25 |
| Vector | Int\* | Vector | 26 |
| Int\* | Vector | Vector | 27 |
| Vector | Vector | Vector | 28 |