УТВЕРЖДЕН

БЮЛИ.00073-01 35 01-ЛУ

ОБ ИЗМЕНЕНИИ НЕ СООБЩАЕТСЯ

СПЕЦИАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «ГАММА СВАЗ-01»

Описание языка

БЮЛИ.00073-01 35 01

Листов 21

КОПИЯ ВЕРНА

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
А-6512	13.12.2023			

2023 Литера О1

АННОТАЦИЯ

Данный документ является описанием языка программирования, на котором разработано специальное программное обеспечение «Гамма СВАЗ-01» (далее по тексту – программа или изделие).

Документ содержит в себе сведения, необходимые для понимания принципов построения программ на языке C++ в среде разработки Qt Creator.

Документ оформлен в соответствии с требованиями ГОСТ 19.506-78.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Программа разработана на языке программирования C++ в среде разработки Qt Creator версии 5.13.1.

1.2. Язык программирования C++ – компилируемый статически типизированный язык программирования общего назначения. Поддерживая разные парадигмы программирования, сочетает свойства как высокоуровневых, так и низкоуровневых языков. Являясь одним из самых популярных языков программирования, C++ широко используется для разработки программного обеспечения. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений.

1.3. Qt Creator предоставляет собой кроссплатформенную, полностью интегрированную среду разработки для создания программ. Она предлагает интеллектуальное завершение кода, подсветку синтаксиса, интегрированную справочную систему, отладчик, а также интеграцию для всех основных систем управления версиями.

2. ЭЛЕМЕНТЫ ЯЗЫКА

2.1. Основные элементы языка C++

2.1.1. Под элементами языка понимают его базовые конструкции, используемые при написании программ. К ним относятся: алфавит, правила записи констант и идентификаторов, основные типы данных и действия над ними.

2.2. Алфавит

2.2.1. Алфавитом называют совокупность символов, используемых в языке. В C++ они образуют буквы, цифры и специальные символы. В качестве букв используются прописные буквы латинского алфавита от A до Z и строчные от a до z, а также знак подчеркивания _. В качестве десятичных цифр используются арабские цифры от 0 до 9. Специальные символы в языке C++ применяются для различных целей: от организации текста программы до определения указаний компилятору языка C++. Специальные символы перечислены в таблице 1.

Таблица 1 – Специальные символы

Символ	Наименование
,	Запятая
.	Точка
;	Точка с запятой
:	Двоеточие
<	Меньше
>	Больше
[	Левая квадратная скобка
]	Правая квадратная скобка
?	Знак вопроса
‘	Одиночная кавычка
(	Левая круглая скобка
)	Правая круглая скобка
{	Левая фигурная скобка
}	Правая фигурная скобка
!	Восклицательный знак
|	Вертикальная черта
/	Наклонная черта вправо (прямой слеш)

Символ	Наименование
\	Наклонная черта влево (обратный слеш)
~	Тильда
#	Знак номера
%	Процент
&	Амперсант
^	Крышка
-	Минус
=	Знак равенства
+	Плюс
*	Звездочка
“	Двойная кавычка

2.2.2. Из символов алфавита формируются лексемы языка:

– идентификаторы;

– знаки операций;

– константы;

– разделители.

2.3. Идентификаторы

2.3.1. Идентификаторы используются как имена переменных, функций и типов данных. Они записываются по следующим правилам:

– идентификаторы начинаются с буквы (знак подчеркивания также является буквой);

– идентификатор может состоять из латинских букв и цифр (пробелы, точки и другие специальные символы при написании идентификаторов недопустимы);

– между двумя идентификаторами должен быть, по крайней мере, хотя бы один пробел;

– идентификатор может быть произвольной длины, но значимыми являются только первые 32 символа в среде WINDOWS и 8 символов в среде DOS; остальные символы игнорируются.

2.3.2. При написании идентификаторов можно использовать как прописные,

так и строчные буквы. В отличие от других языков программирования, компилятор языка С++ различает их в записи идентификатора. Для более простого чтения и понимания идентификаторов рекомендуется использовать имена идентификаторов, состоящие из строчных и прописных букв. Каждый идентификатор имеет тип, который устанавливается при его объявлении. Компилятор языка С++ не допускает использование идентификаторов, совпадающих по написанию с ключевыми словами. Например, идентификатор auto недопустим (однако допустим идентификатор AUTO). Примеры написания идентификаторов приведены в таблице 2.

Таблица 2 – Примеры написания идентификаторов

Правильные идентификаторы	Неправильные идентификаторы
wiggly	$A^**
cat	don't
HOT_key	HOT-key
_grab1	1grab

2.4. Переменные и константы

2.4.1. Данные, обрабатываемые компилятором, – это переменные и константы. Переменные – это данные, которые могут изменять свои значения в процессе выполнения программы. Все переменные в языке С++ должны быть описаны явно. Это означает, что, во-первых, в начале каждой программы или функции Вы должны привести список имен (идентификаторов) всех используемых переменных, а во-вторых, указать тип каждой из них. Оператор описания состоит из спецификации типа и списка имен переменных, разделенных запятой. В конце обязательно должна стоять точка с запятой. При описании возможно задание начального значения переменной. Имя переменной – любая последовательность прописных и строчных букв английского алфавита, цифр и символа подчеркивания '_'. Имя должно начинаться с буквы или символа подчеркивания. Имя может быть произвольной длины, но значимыми являются только первые 32 символа; остальные символы имени игнорируются.

2.4.2. Спецификация типа формируется из ключевых слов, указывающих на различные типы данных. Основные типы в С++ подразделяются на две группы:

целочисленные типы и типы с плавающей точкой. К целочисленным типам относятся типы, представленные следующими именами основных типов: char, short, int, long. Имена целочисленных типов могут использоваться в сочетании с парой модификаторов типа signed и unsigned. Эти модификаторы изменяют формат представления данных, но не влияют на размеры выделяемых областей памяти. Модификатор типа signed указывает, что переменная может принимать как положительные, так и отрицательные значения. Модификатор типа unsigned указывает, что переменная принимает только положительные значения. Основные характеристики целочисленных типов выглядят следующим образом (таблица 3).

Таблица 3 – Основные характеристики целочисленных типов

Тип данных	Байты	Биты	Min	Max
signed char	1	8	- 128	127
unsigned char	1	8	0	255
signed short	2	16	-32768	32767
unsigned short	2	16	0	65535
signed int	2	16	-32768	32767
unsigned int	2	16	0	65535
signed long	4	32	-2147483648	2147483647
unsigned long	4	32	0	4294967295

2.4.3. По умолчанию считается, что данные типов char, int, short int, long используются со знаком. Поэтому ключевое слово signed можно не указывать. Данные типа char используются для хранения символов. Под символом подразумевается одиночная буква, цифра или знак, занимающий только один байт памяти. Переменные типа char могут использоваться как данные со знаком (signed char) и как данные без знака (unsigned char). Если тип char рассматривается как signed, то старший байт его кода определяет знак. В этом случае диапазон значений типа char – от –128 до +127. В случае unsigned char все восемь бит рассматриваются как код, а диапазон возможных значений – от 0 до 255.

2.4.4. К плавающим типам относятся три типа, представленные следующими именами типов, модификаторов и их сочетаний: float, double, long double. Они

используются для работы с вещественными числами, которые представляются в форме записи с десятичной точкой и в "научной нотации". Разница между нотациями становится очевидной из простого примера, который демонстрирует запись одного и того же вещественного числа в различных нотациях:

– 297.7;

– 2.977E2;

– 0.002355;

– 2.355E-3.

2.4.5. В научной нотации слева от символа E записывается мантисса, справа – значение экспоненты, которая всегда равняется показателю степени 10. Ниже представлены основные характеристики типов данных с плавающей точкой (таблица 4).

Таблица 4 – Основные характеристики типов данных с плавающей точкой

Тип данных	Байты	Биты	Min	Max
float	4	32	3.4E-38	3.4E+38
double	8	64	1.7E-308	1.7E+308
long double	10	80	3.4E-4932	3.4E+4932

2.4.6. Константы – это данные, которые устанавливаются равными определенным значениям еще до выполнения программы и сохраняют их на протяжении выполнения всей программы. Имеется четыре типа констант: целые, с плавающей запятой, символьные и перечисляемые. Например, 25 и -5 – целые константы, 4.8, 5E15, -5.1E8 – константы с плавающей запятой, ‘A’, ‘\0’, ‘\n’,’007’ – символьные константы. Набор символов внутри двойных кавычек: “Это строка ”– есть строковая константа. Целые константы могут быть десятичные, восьмеричные и шестнадцатеричные.

2.4.7. Десятичные константы могут принимать значения от 0 до 4.294.967.295. Константы, выходящие за указанные пределы, вызывают ошибку. Отрицательные десятичные константы – это просто константы без знака, к которым применена унарная операция минус.

2.4.8. Восьмеричные константы начинаются с символа нуля, после которого следуют восьмеричные цифры (от 0 до 7), например, 032. Если восьмеричная константа содержит недопустимые цифры 8 или 9, выдается сообщение об ошибке. Ошибка будет также выдаваться при превышении восьмеричной константой значения 037777777777.

2.4.9. Шестнадцатеричные константы начинаются с 0x (или 0X). Значения шестнадцатеричных констант, превышающие 0xFFFFFFFF, приводят к ошибке.

2.4.10. Символьная константа– этоодин или более символов, заключенных в одинарные кавычки, например 'F','=', '\n'. В C++ константа из одного символа имеет тип char.

2.4.11. Для введения управляющих последовательностей, позволяющих получить визуальное представление некоторых, не имеющих графического аналога символов, используется группа символьных констант, которые начинаются со специального символа обратной косой черты (‘\’).

2.4.12. Строковые константы образуют специальную категорию констант, используемых для работы с фиксированными последовательностями символов. Строковая константа имеет тип данных array of char и записывается как последовательность произвольного количества символов, заключенных в двойные кавычки: "Это строковая константа!". Нулевая (пустая) строка записывается как " ".

2.4.13. Перечисляемые константы представляют собой идентификаторы, определенные в объявлениях типа enum. Эти идентификаторы обычно выбираются как мнемонические обозначения для удобства обращения с данными. Перечисляемые константы имеют целочисленный тип данных. Они могут быть использованы в любых выражениях, в которых допустим целочисленный тип данных. Используемые идентификаторы должны быть уникальными в пределах контекста объявления enum. Значения, принимаемые этими константами, зависят от формата объявления перечислимого типа и присутствия опциональных инициализаторов. Например, в операторе enum color { red, yellow, green }; объявляется переменная с именем color, которая может принимать константные значения red, yellow или green.

3. СРЕДСТВА ОБМЕНА ДАННЫМИ

3.1. В C++ обмен данными осуществляется при помощи присваиваний и вызовов функций. Присваивания позволяют возвращать выходные значения функций, либо использовать результаты вычислений выражений в последующих функциях.

4. ВСТРОЕННЫЕ ЭЛЕМЕНТЫ

4.1. Встроенные типы C++

Встроенные типы (также называемые фундаментальными типами) задаются стандартом языка C++ и встроены в компилятор. Встроенные типы не определены ни в одном файле заголовка. Встроенные типы делятся на три main категории: целочисленный, с плавающей запятой и void. Целочисленные типы представляют целые числа. Типы с плавающей запятой могут указывать значения, которые могут иметь дробные части. Большинство встроенных типов рассматриваются компилятором как отдельные типы. Однако некоторые типы являются синонимами или рассматриваются компилятором как эквивалентные типы.

4.1.1. Тип void

4.1.1.1. Тип void описывает пустой набор значений. Переменная типа void не может быть указана. Тип void используется в основном для объявления функций, которые не возвращают значения, или для объявления универсальных указателей на нетипизированные или произвольно типизированные данные. Любое выражение можно явно преобразовать или привести к типу void. Однако такие выражения можно использовать только в следующих операторах и операндах:

– в операторе выражения;

– в левом операнде оператора запятой;

– во втором и третьем операндах условного оператора (? :). std::nullptr_t.

4.1.1.2. Ключевое слово nullptr является константой типа null-указателя, которая может быть преобразована std::nullptr_t в любой необработанный тип указателя.

4.1.2. Тип Boolean

4.1.2.1. Тип bool может иметь значения true и false. Размер bool типа зависит от реализации.

4.1.3. Символьные типы

4.1.3.1. Тип char — это тип представления символов, который эффективно

кодирует элементы базового набора символов выполнения. Компилятор C++ обрабатывает переменные типа char, signed charи unsigned char как переменные разных типов.

4.1.3.2. Переменные типа char по умолчанию повышаются до int типа , как будто из типа signed char , если /J не используется параметр компиляции. В этом случае они рассматриваются как тип unsigned char и повышаются до int без расширения знака.

4.1.3.3. Переменная типа wchar_t — это расширенный или многобайтовый символьный тип L .

4.1.3.4. По умолчанию wchar_t является собственным типом, но может использоваться /Zc:wchar_t- для создания wchar_t определения типа для unsigned short. __wchar_t — синоним для машинного типа wchar_t для систем Майкрософт.

4.1.3.5. Тип char8_t используется для представления символов UTF-8. Он имеет то же представление, что unsigned charи , но рассматривается компилятором как отдельный тип. Тип char8_t является новым в C++20. Для использования требуется параметр компилятора char8_t/std:c++20 или более поздней версии (например /std:c++latest,).

4.1.3.6. Тип char16_t используется для представления символов UTF-16. Он должен быть достаточно большим, чтобы представлять любую единицу кода UTF-16. Компилятор рассматривает его как отдельный тип.

4.1.3.7. Тип char32_t используется для представления символов UTF-32. Он должен быть достаточно большим, чтобы представлять любую единицу кода UTF-32. Компилятор рассматривает его как отдельный тип.

4.1.4. Типы с плавающей запятой

4.1.4.1. Типы с плавающей запятой (таблица 5) используют представление IEEE-754 для обеспечения приближения дробных значений в широком диапазоне величин. В следующей таблице перечислены типы с плавающей запятой в C++ и сравнительные ограничения на размеры типов с плавающей запятой. Эти ограничения требуются стандартом C++ и не зависят от реализации Майкрософт. Абсолютный размер встроенных типов с плавающей запятой не указан в стандарте.

Таблица 5 – Типы с плавающей запятой

Тип	Содержимое
float	Тип float является наименьшим типом с плавающей запятой в C++.
double	double — это тип с плавающей запятой, размер которого больше или равен размеру типа float, но меньше или равен размеру типа long double.
long double	long double — это тип с плавающей запятой, размер которого больше или равен размеру типа double.

4.1.4.2. Представление long double и double идентично. long double Однако и double рассматриваются компилятором как отдельные типы. Компилятор Microsoft C++ использует 4- и 8-байтовые представления IEEE-754 с плавающей запятой.

4.1.5. Целочисленные типы

4.1.5.1. Тип int является базовым целочисленным типом по умолчанию. Он может представлять все целые числа в диапазоне, относящегося к конкретной реализации.

4.1.5.2. Целочисленное представление со знаком может содержать как положительные, так и отрицательные значения. Он используется по умолчанию или при наличии модификатора signed ключевое слово. Модификатор unsigned ключевое слово указывает представление без знака, которое может содержать только неотрицательных значений.

4.1.5.3. Модификатор размера задает ширину используемого целочисленного представления в битах. Язык поддерживает shortмодификаторы , longи long long . Тип short должен быть не менее 16 бит в ширину. Тип long должен быть не менее 32 бит в ширину. Тип long long должен быть не менее 64 бит в ширину. Стандарт задает связь размера между целочисленными типами:

1 == sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long) <= sizeof(long long)

4.1.5.4. Реализация должна поддерживать как минимальные требования к размеру, так и отношение размера для каждого типа. Однако фактические размеры могут отличаться в разных реализациях.

4.1.5.5. При int указании модификаторов размера , ключевое слово может быть опущено signedunsigned. Модификаторы и int тип, если они присутствуют, могут

отображаться в любом порядке. Например, short unsigned и unsigned int short ссылаются на тот же тип.

4.1.6. Синонимы целочисленных типов

4.1.6.1. Следующие группы типов считаются синонимами компилятором:

- short, short int, signed short, signed short int
- unsigned short, unsigned short int
- int, signed, signed int
- unsigned, unsigned int
- long, long int, signed long, signed long int
- unsigned long, unsigned long int
- long long, long long int, signed long long, signed long long int
- unsigned long long, unsigned long long int

4.1.6.2. Типы целых чисел, относящиеся к корпорации Майкрософт, включают типы определенной ширины __int8, __int16, __int32и __int64 . Эти типы могут использовать модификаторы signed и unsigned . Тип данных __int8 аналогичен типу char, __int16 — типу short, __int32 — типу int, а __int64 — типу long long.

4.1.7. Размеры встроенных типов

4.1.7.1. Большинство встроенных типов имеют размеры, определенные реализацией. В таблице 6 приведен объем хранилища, необходимый для встроенных типов в Microsoft C++. В частности, long составляет 4 байта даже в 64-разрядных операционных системах.

Таблица 6 – Размеры встроенных типов

Тип	Размер
bool, char, char8_t, unsigned char, signed char, __int8	1 байт
char16_t, __int16, short, unsigned short, wchar_t, __wchar_t	2 байта
char32_t, float, __int32, int, unsigned int, long, unsigned long	4 байта
double, __int64, long double, long long, unsigned long long	8 байт

4.2. Встроенные операторы C++

4.2.1. Арифметические операторы

4.2.1.1. Арифметические операторы, поддерживаемые языком C ++, приведены в таблице 7.

Таблица 7 – Арифметические операторы

Оператор	Описание	Пример
+	Добавляет два операнда	A + B даст 30
-	Вычитает второй операнд с первого	A - B даст -10
*	Умножает оба операнда	A * B даст 200
/	Делит числитель на де-числитель	B / A даст 2
%	Оператор модуля и остаток после целочисленного деления	B% A даст 0
++	Оператор приращения увеличивает целочисленное значение	A ++ даст 11
-	Уменьшает целочисленное значение на единицу	A-- даст 9

4.2.2. Реляционные операторы

4.2.2.1. Реляционные операторы, поддерживаемые языком C ++, приведены в таблице 8.

Таблица 8 – Реляционные операторы

Оператор	Описание	Пример
==	Проверяет, равны ли значения двух операндов или нет, если да, то условие становится истинным.	(A == B) не соответствует действительности.
знак равно	Проверяет, равны ли значения двух операндов или нет, если значения не равны, условие становится истинным.	(A! = B) истинно.
>	Проверяет, превышает ли значение левого операнда значение правого операнда, если да, тогда условие становится истинным.	(A> B) неверно.
<	Проверяет, является ли значение левого операнда меньше значения правильного операнда, если да, тогда условие становится истинным.	(A <B) истинно.
> =	Проверяет, превышает ли значение левого операнда значение правого операнда, если да, тогда условие становится истинным.	(A> = B) неверно.

Оператор	Описание	Пример
<=	Проверяет, является ли значение левого операнда меньше или равно значению правильного операнда, если да, тогда условие становится истинным.	(A <= B) истинно.

4.2.3. Логические операторы

4.2.3.1. Логические операторы, поддерживаемые языком C ++, приведены в таблице 9.

Таблица 9 – Логические операторы

Оператор	Описание	Пример
&&	Вызывается логическим оператором AND. Если оба операнда отличны от нуля, условие становится истинным.	(A && B) является ложным.
||	Вызывается логическим оператором ИЛИ. Если любой из двух операндов отличен от нуля, тогда условие становится истинным.	(A || B) истинно.
!	Вызывается логическим оператором NOT. Используется для изменения логического состояния операнда. Если условие истинно, то логический оператор NOT сделает ложным.	! (A && B) истинно.

4.2.4. Побитовые операторы

4.2.4.1. Побитовый оператор работает с битами и выполняет побитовую операцию. Таблица истинности для &, |, и ^ приведена в таблице 10.

Таблица 10 – Таблица истинности для побитовых операторов

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

4.2.4.2. Побитовые операторы, поддерживаемые языком C ++, приведены в таблице 11.

Таблица 11 – Побитовые операторы

Оператор	Описание	Пример
&	Двоичный оператор AND копирует бит в результат, если он существует в обоих операндах.	(A & B) даст 12, что составляет 0000 1100
|	Двоичный оператор OR копирует бит, если он существует в любом из операндов.	(A | B) даст 61, который равен 0011 1101
^	Оператор двоичного XOR копирует бит, если он установлен в один операнд, но не тот и другой.	(A ^ B) даст 49, который равен 0011 0001
~	Binary Ones Оператор дополнения является унарным и имеет эффект «flipping» бит.	(~ A) даст -61, что составляет 1100 0011 в форме дополнения 2 из-за подписанного двоичного числа.
<<	Двойной левый оператор сдвига.Значение левых операндов перемещается влево на количество бит, заданных правым операндом.	A << 2 даст 240, что составляет 1111 0000
>>	Двоичный оператор правого сдвига. Значение левых операндов перемещается вправо на количество бит, заданных правым операндом.	A >> 2 даст 15, что составляет 0000 1111

4.2.5. Операторы присваивания

4.2.5.1. Операторы присваивания, поддерживаемые языком C ++, приведены в таблице 12.

Таблица 12 – Операторы присваивания

Оператор	Описание	Пример
знак равно	Простой оператор присваивания, присваивает значения из правых операндов в левый операнд.	C = A + B присваивает значение A + B в C
+ =	Оператор Add AND присваивания, Он добавляет правый операнд в левый операнд и присваивает результат левому операнду.	C + = A эквивалентно C = C + A
знак равно	Subtract AND assign operator, вычитает правый операнд из левого операнда и присваивает результат левому операнду.	C - = A эквивалентно C = C - A
знак равно	Оператор умножения и присваивания, Он умножает правый операнд на левый операнд и присваивает результат левому операнду.	C * = A эквивалентно C = C * A
знак равно	Оператор Divide AND assign. Он делит левый операнд на правый операнд и присваивает результат левому операнду.	C / = A эквивалентно C = C / A

Оператор	Описание	Пример
знак равно	Модуль и оператор присваивания, он принимает модуль с использованием двух операндов и присваивает результат левому операнду.	C% = A эквивалентно C = C% A
<< =	Оператор сдвига слева и.	C << = 2 совпадает с C = C << 2
>> =	Оператор правой смещения и назначения.	C >> = 2 совпадает с C = C >> 2
знак равно	Побитовый И оператор присваивания.	C & = 2 является таким же, как C = C & 2
^ =	Побитовое исключающее ИЛИ и оператор присваивания.	C ^ = 2 является таким же, как C = C ^ 2
| =	Побитовое включение оператора OR и присваивания.	C | = 2 совпадает с C = C |2

4.2.6. Другие операторы

4.2.6.1. В таблице 13 перечислены некоторые другие операторы, поддерживаемые C ++.

Таблица 13 – Другие операторы, поддерживаемые C ++

Оператор	Описание
sizeof	Возвращает размер переменной. Например, sizeof (a), где 'a' является целым числом и будет возвращать 4.
Condition ? X : Y	Если Условие истинно, то оно возвращает значение X, иначе возвращает значение Y.
,	Вызывает последовательность операций. Значение всего выражения запятой - это значение последнего выражения списка, разделенного запятыми.
. (dot) and -> (arrow)	Используются для ссылки на отдельных членов классов, структур и союзов.
Cast	Преобразуют один тип данных в другой. Например, int (2.2000) вернет 2.
&	Возвращает адрес переменной. Например, & a; даст фактический адрес переменной.
*	Является указателем на переменную. Например * var; будет указывать на переменную var.

4.2.7. Приоритеты операторов в C ++

4.2.7.1. Приоритет оператора определяет группировку терминов в

выражении. Это влияет на оценку выражения. Некоторые операторы имеют более высокий приоритет, чем другие. Например, оператор умножения имеет более высокий приоритет, чем оператор сложения. Например, x = 7 + 3 * 2; здесь x назначается 13, а не 20, потому что оператор * имеет более высокий приоритет, чем +, поэтому сначала умножается 3 * 2, а затем добавляется 7.

4.2.7.2. В таблице 14 операторы с наивысшим приоритетом появляются в верхней части таблицы, а нижние - внизу. Внутри выражения сначала будут оцениваться операторы с более высоким приоритетом.

Таблица 14 – Приоритеты операторов

Категория	Оператор	Ассоциативность
постфикс	() [] ->. ++ - -	Слева направо
Одинарный	+ -! ~ ++ - - (тип) * & sizeof	Справа налево
Multiplicative	* /%	Слева направо
присадка	+ -	Слева направо
сдвиг	<< >>	Слева направо
реляционный	<<=>> =	Слева направо
равенство	==! =	Слева направо
Побитовое AND	&	Слева направо
Побитовое XOR	^	Слева направо
Побитовое OR	|	Слева направо
Логические AND	&&	Слева направо
Логический OR	||	Слева направо
условный	?:	Справа налево
присваивание	= + = - = * = / =% = >> = << = & = ^ = | =	Справа налево
запятая	,	Слева направо

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Входящий номер сопроводительного документа и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					