2.2 Вопросы:

1) Что такое константа? Чем отличаются константы compile time и runtime?

Константы — переменные, значения которых изменить нельзя. В C++ есть два вида констант по времени выполнения.

Константы по времени выполнения подразделяются:

- compile time (время компиляции)
- runtime (временем выполнения программы).

2) Чем отличаются литеральные константы от символьных?

Литеральные константы (обычно просто литералы) – это значения, которые вставляются непосредственно в код. Они являются константами, так как изменить их значения нельзя.

Символьная константа — это тот же литерал (магическое число), только с идентификатором.

3) Какими способами можно задать символьную константу в C++?

Конструкция (более предпочтительна для чтения const перед типом данных):

const <тип> <ИмяПеременной> = Значение;

пример: const double gravity { 9.8 };

constexp <тип> <ИмяПеременной> = Значение;

пример: constexpr double quadro (9.8);

4) Как реализуется задание константы с помощью макросов?

Конструкция Макросы-объекты:

#define <Идентификатор>

#define <Идентификатор> <Значение/Текст(Символ)/Литерал>

5) Что такое перечисляемый тип данных enum? Чем он отличается от класса enum?

Перечисляемый тип — это тип данных, где любое значение определяется как символьная константа.

класс enum (или перечисление с областью видимости) - добавляет перечислению, ***локальную область видимости с её правилами и строгие правила типов***. Являются более безопасными в использовании

6) Конвертируйте заданное преподавателем положительное десятичное число в двоичную систему счисления. Примеры для тренировки: 93, 56, 254, 999.

Десятичное	Бинарное	Десятичное	Бинарное	Десятичное	Бинарное	Десятичное	Бинарное
93 / 2 = 46	1	56 / 2 = 46	0	254 / 2 = 127	0	999 / 2 = 499	1
46 / 2 = 23	0	28 / 2 = 14	0	127 / 2 = 63	1	499 / 2 = 249	1
23 / 2 = 11	1	14 / 2 = 7	0	63 / 2 = 31	1	249 / 2 = 124	1
11 / 2 = 5	1	7 / 2 = 3	1	31 / 2 = 15	1	124/ 2 = 62	0
5 / 2 = 4	1	3 / 2 = 2	1	15 / 2 = 7	1	62 / 2 = 31	0
4 / 2 = 2	0	2 / 2 = 1	1	7 / 2 = 3	1	31 / 2 = 15	1
2 / 2 = 1	1			3 / 2 = 1	1	15 / 2 = 7	1
				1 / 2 = 0	1	7 / 2 = 3	1
						3 / 2 = 1	1
						1 / 2 = 0	1

93 (десятичное) = 101 1101 (двоичное)

56 (десятичное) = 11 1000 (двоичное)

254 (десятичное) = 1111 1110 (двоичное)

999 (десятичное) = 11 1110 0111 (двоичное)

7) Конвертируйте заданное преподавателем восьмизначное двоичное число в десятичную систему счисления без знака. Примеры для тренировки: 1010 0010, 0010 1111.

Двоичный символ 1 0 1 0 0 0 1 0

* Значение символа 128 64 32 16 8 4 2 1

= Результат (162) 128 0 32 0 0 0 2 0

Двоичный символ 0 0 1 0 1 1 1 1

* Значение символа 128 64 32 16 8 4 2 1

= Результат (47) 0 0 32 0 8 4 2 1

8) Конвертируйте заданное преподавателем отрицательное десятичное число в двоичную систему счисления, используя метод two’s complement. Примеры для тренировки: -93, -56, -254, -999.

-93

Сначала выясняем бинарное представление 93: 0101 1101

Затем инвертируем все биты (конвертируем в противоположные): 1010 0010

Затем добавляем к числу единицу: 1010 0011

-56

Представление положительного 56: 0011 1000

Инвертируем все биты: 1100 0111

Добавляем к числу единицу: 1100 1000

-254

Представление положительного 254: 1111 1110

Инвертируем все биты: 0000 0001

Добавляем к числу единицу: 0000 0010

-999

Представление положительного 999: 0011 1110 0111

Инвертируем все биты: 1100 0001 1000

Добавляем к числу единицу: 1100 0001 1001

9) Конвертируйте заданное преподавателем двоичное число в десятичную систему счисления со знаком, используя метод two’s complement. Примеры для тренировки: 1010 0010, 0010 1111, 1111 0100.

Имеем: 1010 0010

Инвертируем биты: 0101 1101

Добавляем единицу: 0101 1110

Конвертируем в десятичную систему счисления:

(0 * 128) + (1 * 64) + (0 * 32) + (1 * 16) + (1 * 8) + (1 * 4) + (1 * 2) + (0 * 1) = 64 + 16 + 8 + 4 + 2 = 94

Имеем: 0010 1111

Инвертируем биты: 1101 0000

Добавляем единицу: 1101 0001

Конвертируем в десятичную систему счисления:

(1 * 128) + (1 * 64) + (0 * 32) + (1 * 16) + (0 * 8) + (0 * 4) + (0 * 2) + (1 * 1) = 128 + 64 + 32 + 16 + 1 = 241

Имеем: 1111 0100

Инвертируем биты: 0000 1011

Добавляем единицу: 0000 1100

Конвертируем в десятичную систему счисления:

(0 * 128) + (0 * 64) + (0 * 32) + (0 * 16) + (1* 8) + (1 * 4) + (0 * 2) + (0 * 1) = 8 + 4 = 12

10)Как понять, что двоичное число со знаком положительное или отрицательное?

С целыми числами **signed** используется метод «***two's complement***»(дополнительный код числа, или дополнение до двойки (two's complement) это обратный код, к младшему значащему разряду которого прибавлена единица, используется в C++.). Он означает, что самый левый (самый главный) бит используется в качестве знакового бита. Если значением знакового бита является 0, то число положительное, если 1 — число отрицательное.

11)Что такое операторы? Назовите пять любых операторов. Как повысить приоритет выполнения оператора?

Оператор — это символ, который сообщает компилятору выполнить определенные математические или логические манипуляции.

- Арифметические операторы
- Реляционные операторы
- Логические операторы
- Побитовые операторы
- Операторы присваивания

Если у двух операторов в выражении одинаковый уровень приоритета, и они размещены рядом, компилятор будет использовать **правила ассоциативности**, которые указывают направление выполнения операций: слева направо или справа налево.

Например, в выражении 3 * 4 / 2 операции умножения и деления имеют одинаковый уровень приоритета (5-й уровень). А ассоциативность пятого уровня соответствует выполнению операций слева направо, таким образом: (3 * 4) / 2 = 6.

12)Чем отличаются унарные и бинарные операторы?

Унарные операторы – это те, которые применяются только к одному операнду.

Унарный оператор плюс возвращает значение операнда. Унарный оператор минус возвращает

операнд, умноженный на -1.

Бинарные операторы – это те, которые применяются к двум операндам (слева и справа).

Пример:

Оператор Символ Действие Операция

сложение: + x + y; x плюс y.

13)Назовите логические операторы сравнения. Напишите несколько примеров сравнения с логическими операциями.

Логические операторы сравнения:

Оператор Символ

Логическое НЕ !

Логическое И &&

Логическое ИЛИ ||

Пример:

не x и y => !x && y;

1 или 56 => 1 || 56

14)Какой оператор может заменить условие if/else? Как он структурирован?

Условный тернарный оператор - может заменить условие if/else.

Конструкция тернарного оператора:

(условие) ? выражение : другое_выражение;

15)Что такое идентификатор?

Идентификатор — это имя переменной, функции, класса или другого объекта в языке C++. Мы можем определять идентификаторы любыми словами/именами.

16)Что такое побитовые операции? Напишите побитовые операции?

Поразрядные операции выполняются над отдельными разрядами или битами чисел.

Побитовые операторы манипулируют отдельными битами в пределах переменной.

Побитовый сдвиг влево <<

Побитовый сдвиг вправо >>

Побитовое НЕ ~

Побитовое И &

Побитовое ИЛИ |

Побитовое исключающее ИЛИ (XOR) ^

17)Приведите пример побитового сдвига влево, вправо.

Побитовый сдвиг влево (<<):

3 = 00000011

3 << 1 = 0000 0110 = 6

3 << 2 = 0000 1100 = 12

3 << 3 = 0000 1000 = 8

Побитовый сдвиг вправо (>>)

12 = 00001100

12 >> 1 = 00000110 = 6

12 >> 2 = 00000011 = 3

12 >> 3 = 00000001 = 1

18)Как работает побитовое НЕ?

Побитовое НЕ меняет каждый бит на противоположный, например, с 0 на 1 или с 1 на 0.

19)Как работают логические И, ИЛИ и исключающее ИЛИ (XOR)?

Логическое ИЛИ. Если хоть одно из двух условий истинно, то логический оператор ИЛИ – true. Если только левый операнд или только правый, или оба сразу true, результат — true.

Логическое И. Только при условии, что оба операнда будут истинными, логический оператор И будет true. Если нет – false.

Логическое НЕ. Следует помнить, что логический оператор НЕ имеет очень высокий уровень приоритета.

20)Как работают побитовые И, ИЛИ и исключающее ИЛИ (XOR)?

Побитовые И (&) и ИЛИ (|) работают аналогично логическим И/ИЛИ. Однако, они применяются к каждому биту отдельно.

Побитовые ИЛИ А затем применять операцию к каждому столбцу битов. Логическое ИЛИ возвращает true (1), если один из двух или оба операнды истинны

(1), также работает и побитовое ИЛИ.

Побитовое И работает аналогично логическому И: true возвращается, только если оба бита в столбце равны 1.

Побитовое исключающее ИЛИ (^) (на англ. — XOR). При обработке двух операндов, исключающее ИЛИ возвращает true (1), только если один и только один из операндов является истинным (1). Если таких нет или все операнды равны 1, то результатом будет false (0).

21)Каков результат 0110 >> 2 в двоичной системе счисления?

0110 >> 2 = 0001

22)Каков результат 5 | 12 в десятичной системе счисления?

12 = 1100

5 = 0101

Следовательно, 5 | 12 будет:

1100 // 12

0101 // 5

1101 // 13

23)Каков результат 5 & 12 в десятичной системе счисления?

5 = 0101

12 = 1100

Следовательно, 5 & 12 будет:

0101 // 5

1100 // 12

0100 // 4

24)Каков результат 5 ^ 12 в десятичной системе счисления?

5 = 0101

12 = 1100

Следовательно, 5 ^ 12 будет:

0101 // 5

1100 // 12

1001 // 9

25)Каков результат !12 в десятичной системе счисления (пусть выделено 8 бит).

Пусть выделено 8 бит:

12 = 0000 1100

~ 12 = 1111 0011

Имеем: 1111 0011

Инвертируем биты: 0000 1100

Добавляем единицу: 0000 1101

Так как исходный знаковый бит был отрицательным, то результатом является

~ 12 = -13 (десятичное)