Вопросы:

1. Что такое массивы? Какой индекс у первого элемента массива?

Массив — это область памяти, где могут последовательно храниться несколько значений, или это непрерывный участок памяти, содержащий последовательность объектов одинакового типа, обозначаемый одним именем.

Индексирование массива начинается с 0.

2. Что такое многомерные массивы? Сколько измерений может быть в многомерном массиве?

Массив массивов называется многомерным массивом.

В многомерном массиве может быть более двух измерений. Но работать с таким массивом крайне трудно.

3. Как задаются измерения многомерного массива? Какое измерение можно пропускать, и в каком случае?

Трехмерные массивы трудно инициализировать любым интуитивным способом с использованием списка инициализаторов, поэтому лучше инициализировать весь массив значением 0 и явно присваивать значения с помощью вложенных циклов.

Необходимо указывать оба измерения 2D-массивов, за исключением случаев, когда это указано в параметре функции или если присутствует инициализатор, тогда первое измерение может быть опущено.

Первое измерение можно опустить. Но второе измерение должно присутствовать, чтобы сообщить компилятору, что указатель 2D_arr является указателем на массив из n целых чисел. Компилятор использует длину инициализатора только для вычисления первого измерения. Остальная часть измерения должна быть явно указана во время объявления.

4. Как инициализировать двумерный массив? Существуют ли более удобные способы придать значения элементам массивов?

Инициализация двумерного массива:

<тип данных> идентификатор [длинна строк N] [длинна столбцовM] ;

Для инициализации двумерного массива проще всего использовать вложенные фигурные скобки, где каждый набор значений соответствует определенной строке.

Хотя некоторые компиляторы могут позволить вам упустить внутренние фигурные скобки, все же рекомендуется указывать их в любом случае: улучшается читабельность и уменьшается вероятность получения незапланированных нулевых элементов массива из-за того, что C++ заменяет отсутствующие инициализаторы значением 0.

5. Какие циклы Вы знаете? Какой у них синтаксис?

В C++ есть 3 типа циклов:

- while,
- do while
- for
- for each (C++ 11 добавила поддержку еще одного специфического типа циклов

while

Конструкция 1, если одно условие, то можно без блока стейтменов:

while (условие)

statement_1; // если одно условие, то можно без блока стейтменов

Конструкция, если одно условие, то можно без блока стейтменов:

Конструкция 2, больше одного условия, нужно блок стейтменов:

while (условие){

statement_1;

statement_2; / }// больше одного условия, нужно блок стейтменов

do while

Конструкция 1 если одно условие, то можно без блока стейтменов:

do

statement_1;

while (условие);

Конструкция 2, больше одного условия, нужно блок стейтменов:

do {

// больше одного условия, нужно блок стейтменов

statement_1;

statement_2; }

while (условие);

for

Конструкция 1 если одно условие, то можно без блока стейтменов:

for (объявление переменных; условие; инкремент/декремент счетчика)

statement_1; // тело цикла

Конструкция 2, больше одного условия, нужно блок стейтменов:

for (объявление переменных; условие; инкремент/декремент счетчика) {

// больше одного условия, нужно блок стейтменов

statement_1;

statement_2;

}

for each

Конструкция :

variable_name — переменная, которая будет перебирать массив или вектор.

array/vector_name — это имя соответствующего набора данных, по которому будет проходить цикл,

for(type variable_name : array/vector_name)

{

statement_1;

...

}

6. Что такое рекурсия и итерация? Чем они отличаются? Как их применять? Что лучше применять (в большинстве случаев) и почему?

Итерация/перемещение это перемещение по элементам массива (или какой-нибудь другой структуры) с использованием циклов и индексов (циклы for и while), а также с помощью циклов for с явным указанием диапазона.

Рекурсивная функция (или просто «***рекурсия***») в языке C++ — это функция, которая вызывает сама себя.

Итерация

Использование циклов с массивами: Циклы с массивами используются, например, для выполнения задач:

- Вычислить значение (например, среднее или сумму всех значений).
- Найти значение (например, самое высокое или самое низкое).

• Отсортировать элементы массива (например, по возрастанию или по убыванию).

Рекурсия

Типичными рекурсивными задачами являются задачи: нахождения n!, числа Фибоначчи. Всё решение сводится к решению основного или, как ещё его называют, базового случая

Если рекурсивный алгоритм проще реализовать, то имеет смысл начать с рекурсии, а затем уже оптимизировать код в итеративный алгоритм. Рекомендуется использовать итерацию, вместо рекурсии, но в тех случаях, когда это действительно практичнее.

7. Что делает оператор continue? Где он применяется?

Оператор перехода ***continue*** позволяет сразу перейти в конец тела цикла, пропуская весь код, который находится под ним. ***Continue*** пропускает лишь одну итерацию цикла.

8. Что делает оператор break? Где он применяется?

Оператор ***break*** используется в циклах для досрочного выхода из цикла.

9. Как можно генерировать случайные числа в C++? Являются ли случайные числа в действительности случайными? Для чего существует привязка к time(0)? Для чего может понадобиться генерация случайных чисел?

C/C++ также имеет свой собственный встроенный генератор случайных чисел. Он реализован в двух отдельных функциях, которые находятся в заголовочном файле cstdlib:

• srand() – устанавливает стартовое значение, выбранное пользователем. srand() следует вызывать только один раз — в начале программы (обычно в верхней части функции main()).

• rand() — генерирует следующее случайное число в последовательности. Оно будет из диапазона от 0 до RAND_MAX (константа в cstdlib, значение которой — 32767).

Если запустить программу несколько раз, она каждый раз будет выводить одни и те же числа. Это означает, что, хотя каждое число в последовательности кажется случайным относительно предыдущего, вся последовательность не является случайной вообще!

В языке Си есть функция time(), которая возвращает в качестве времени общее количество секунд, прошедшее от полуночи 1 января 1970 года.

(time(0)) - устанавливает значение системных часов в качестве стартового числа.

Возможность генерировать случайные числа очень полезна в некоторых видах программ, в частности, в играх, программах научного или статистического моделирования.

10. Что такое сортировка массива? Зачем она может быть нужна? Какие способы реализации сортировки в C++ Вы знаете?

Сортировка массива — это процесс распределения всех элементов массива в определенном порядке.

Очень часто применяется при обработке данных. Сортировка данных может сделать поиск внутри массива более эффективным не только для людей, но и для компьютеров.

Сортировка массивов методом выбора, пузырька.

11. Многофайловые программы зачем нужны (ну и было бы все в одном файле)?

Разделение исходного текста программы на несколько файлов становится необходимым по многим причинам:

1. С большим текстом просто неудобно работать.

2. Разделение программы на отдельные модули, которые решают конкретные подзадачи.

3. Разделение программы на отдельные модули, с целью повторного использования этих модулей в других программах.

4. Разделение интерфейса и реализации.

12. Что такое заголовочный файл *.h, *.hpp предназначение, содержание?

Заголовочный файл (файл заголовка, подключаемый файл или header file). Они имеют расширение .h, но иногда их можно увидеть и с расширением .hpp или вообще без расширения. Целью заголовочных файлов является удобное хранение предварительных объявлений для использования другими файлами.

Заголовочные файлы состоят из двух частей:

- Дерективы процессора — в частности header guard. Header guards предотвращают вызовы (#include) заголовочного файла больше одного раза с одного и того же файла (рассмотрим ниже).

- Содержимое заголовочного файла – сам контент файла .h. В нем находятся предварительные объявления всех функций, переменных, классов, структур, которые мы будем в дальнейшем использовать. Все заголовочные файлы должны быть с расширением .h.

13. Что такое файл реализации *.cpp, назначение, состав?

Файл реализации — .cpp для программ на C++ и .c, для программ на языке C. (Хотя в STL включаемые файлы вообще без расширений, но, по сути, они являются заголовочными файлами.)

Реализация(*.cpp):

Правило 1. Что может быть в файле реализации Файл реализации может содержать как определения, так и объявления. Объявления, сделанные в файле реализации, будут лексически локальны для этого файла. Т.е. будут действовать только для этой единицы компиляции.

14. Что такое Header guards, #pragma once?

header guards (“охранники заголовков”, другое название — include guards). Header guards — это директивы условной компиляции.

Некоторые компиляторы поддерживают более простую, альтернативную форму header guards — директиву #pragma (Ms VisualStudio)

#pragma once используется, как и header guards, но имеет дополнительные преимущества – она короче и менее подвержена ошибкам.

15. Что такое неявное приведение типов данных?

Неявное преобразование типа (также называемое автоматическим преобразованием типа или принудительным) выполняется всякий раз, когда операнды имеют различные типы, и пользователь не указывает компилятору, как выполнить конвертацию.

16. Что такое числовое расширение?

Числовым расширением (или продвижением) - когда значение одного типа данных преобразовывается в значение другого похожего типа данных, но более крупного.

```
int main() {
        long l(65); // расширяем значение типа int (65) в тип
        long double d(0.11f); // расширяем значение типа float (0.11) в тип double
        return 0;
}
```

17. Что такое числовая конверсия?

Числовой конверсией - преобразование значение из более крупного типа данных в аналогичный более мелкий тип или конвертация происходит между разными типами.

Пример:

```
int main() {
        double d = 4; // конвертируем 4 (тип int) в
        double short s = 3; // конвертируем 3 (тип int) в short
        return 0;
}
```

18. Что такое обработка арифметических выражений?

При обработке выражений компилятор разбивает каждое выражение на отдельные подвыражения. Арифметические операторы требуют, чтобы их операнды были одного типа. Чтобы это гарантировать, компилятор использует следующие правила:

- Если операнд — целое число типа меньше int, то оно подвергается интегральному расширению в int или в unsigned int.

• Если операнды все еще не подходят, тогда компилятор вычисляет операнд с наивысшим приоритетом и неявно преобразовывает тип другого операнда в соответствие к типу первого. Приоритет типов операндов следующий:

• long double (самый высокий);

• double;

• float;

• unsigned long long;

• long long;

• unsigned long;

• long;

• unsigned int;

• int (самый низкий).

цепочки преобразований:

bool -> char -> short -> int -> double -> long double

bool -> char -> short -> int -> long -> long long

unsigned char -> unsigned short -> unsigned int -> unsigned

long float -> double -> long double

19. Что такое явное приведение типов данных?

Явное приведение задаётся программистом в тексте программы, чтобы указать компилятору выполнить явное преобразование. В языке C++ есть 5 видов операций явного преобразования типов cast:

• C-style cast;

• const cast;

• static_cast;

• dynamic cast;

• reinterpret cast.

Приведения типов данных может быть использован для преобразования любого типа в любой другой тип.

20. Что делает оператор C-style cast (int)?

Наследие от языка C, явное преобразование типов данных выполняется с помощью оператора () – внутри пишется тип, в который нужно конвертировать.

cast (int) – конвертирует в целочисленное значение тип ***integer.***

21. Что делает оператор static_cast?

static_cast — унарная операция приведения типов данных. static_cast может быть использована для преобразования одного типа в другой, но она не должна быть использована для выполнения недопустимого преобразования, например, преобразование значения в указатель или наоборот. Основным преимуществом static_cast является проверка компилятором во время компиляции, что усложняет возможность возникновения непреднамеренных ошибок. static_cast также (специально) имеет меньшее влияние, чем C□style cast, поэтому вы не сможете случайно изменить const или сделать другие вещи, которые не намеревались делать.