

Какие утверждения о `const` и указателях верны?

Выберите один или несколько ответов:

- ☒ a. `const int* const p` — константный указатель на константу
- ☒ b. `int* const p` — константный указатель
- ☒ c. `const_cast` может удалять константность с undefined behavior
- ☒ d. `const int* p` — указатель на константу

Что такое "dangling pointer"?

Выберите один или несколько ответов:

- ☐ a. Неинициализированный указатель
- ☒ b. Указатель на освобожденную память
- ☐ c. Указатель на константный объект
- ☒ d. Указатель, инвалидированный сдвигом базового адреса

Как ведет себя `reinterpret_cast` при преобразовании указателей?

Выберите один или несколько ответов:

- ☒ a. Корректно конвертирует между указателями на несвязанные типы
- ☒ b. Сохраняет адрес без изменения битового представления
- ☒ c. Может нарушать strict aliasing rules
- ☐ d. Гарантирует выравнивание для целевого типа

Что верно о `std::atomic`?

Выберите один или несколько ответов:

- ☐ a. Эквивалентен `volatile T*`
- ☒ b. Поддерживает арифметику указателей
- ☒ c. Гарантирует атомарность операций присваивания
- ☒ d. Требует lock-free реализации

Что такое "strict aliasing"?

Выберите один или несколько ответов:

- ☐ a. Гарантия выравнивания указателей
- ☒ b. Правило оптимизации компилятора
- ☒ c. Запрет доступа к объекту через указатель несовместимого типа
- ☒ d. Исключение для `char*` и `std::byte*`

Какие операции с указателями приводят к UB?

Выберите один или несколько ответов:

- ☐ a. Преобразование `int` в указатель
- ☒ b. Сравнение указателей из разных массивов
- ☒ c. Выход за границы массива
- ☒ d. Разыменование `nullptr`

Что верно об арифметике указателей?

Выберите один ответ:

- ☐ a. `ptr - nullptr` определено стандартом
- ☐ b. Разность двух указателей возвращает количество байт между ними
- ☒ c. Указатели разных типов нельзя сравнивать
- ☐ d. `ptr + n` эквивалентно `(char*)ptr + n * sizeof(*ptr)`

[Очистить мой выбор](#)

Что гарантирует `std::unique_ptr`?

Выберите один или несколько ответов:

- ☐ a. Возможность копирования без ограничений
- ☒ b. Совместимость с массивами через `std::unique_ptr`
- ☐ c. Потокобезопасность операций
- ☒ d. Автоматическое освобождение памяти при выходе из области видимости

Как динамически выделить массив с инициализацией?

Выберите один или несколько ответов:

- ☒ a. `new int[10]{1,2}`
- ☐ b. `malloc(sizeof(int)*10)`
- ☒ c. `new int[10]()`
- ☐ d. `std::make_unique(10)`

Какие конструкции эквивалентны для доступа к элементу массива?

Выберите один или несколько ответов:

- ☒ a. `arr[i]`
- ☒ b. `*(arr + i)`
- ☐ c. `*arr + i`
- ☒ d. `i[arr]`

Что из перечисленного инвалидирует указатель?

Выберите один или несколько ответов:

- ☒ a. Выход объекта из области видимости (stack memory)
- ☒ b. Освобождение памяти через `delete`
- ☒ c. Сброс `std::shared_ptr`, владеющего объектом
- ☐ d. Модификация константного объекта через указатель

Какие операции разрешены с указателем типа `void*`?

Выберите один или несколько ответов:

- ☐ a. Разыменование
- ☐ b. Арифметика указателей (инкремент/декремент)
- ☒ c. Приведение к типизированному указателю
- ☒ d. Сравнение с `nullptr`

Какие ошибки выявляет `std::shared_ptr`?

Выберите один или несколько ответов:

- ☐ a. Утечки памяти при циклических ссылках
- ☐ b. Неправильный контроль владения
- ☒ c. Разыменование после удаления объекта
- ☒ d. Двойное освобождение памяти

Что делает `std::weak_ptr` полезным?

Выберите один или несколько ответов:

- ☒ a. Наблюдение за объектом без владения
- ☒ b. Предотвращение циклических ссылок
- ☐ c. Автоматическое освобождение ресурсов при нулевом счетчике
- ☒ d. Доступ к объекту только при существовании `shared_ptr`

Какие свойства указателей проверяются в `constexpr`?

Выберите один или несколько ответов:

- ☒ a. Адреса статических объектов
- ☒ b. Результаты арифметики указателей
- ☒ c. Разыменование в `compile-time`
- ☐ d. Валидность `reinterpret_cast`

Какие утверждения о размере указателей верны?

Выберите один или несколько ответов:

- ☐ a. Размер указателя зависит от типа данных, на который он указывает
- ☒ b. `sizeof(int*)` равен `sizeof(char*)` на одной платформе
- ☐ c. `sizeof(void*)` определяет размер адресного пространства
- ☐ d. Указатели на функции имеют тот же размер, что и указатели на данные

Что верно о связи массивов и указателей?

Выберите один или несколько ответов:

- ☒ a. `sizeof(arr)` возвращает размер массива в байтах
- ☒ b. `&arr[0]` идентичен `&arr` по значению
- ☒ c. Многомерные массивы хранятся в row-major порядке
- ☐ d. `int arr[5];` — `arr` является lvalue, кроме контекста decay

Какие действия с `std::shared_ptr` изменяют счетчик ссылок?

Выберите один или несколько ответов:

- ☒ a. Копирование через конструктор копирования
- ☐ b. Перемещение через `std::move`
- ☐ c. Создание `std::weak_ptr` из `shared_ptr`
- ☒ d. Вызов `reset()`

Укажите корректные способы объявления указателя на константный массив:

Выберите один или несколько ответов:

- ☒ a. `const int (*arr)[10]`
- ☐ b. `int (* const arr)[10]`
- ☐ c. `int const * arr[10]`
- ☐ d. `const int* arr[10]`

Что верно о "pointer to function"?

Выберите один или несколько ответов:

- ☒ a. Вызов через указатель эквивалентен прямому вызову
- ☐ b. Может указывать на шаблон функции
- ☒ c. Тип `void (*)(int)` отличается от `void (&)(int)`
- ☒ d. Совместим с лямбда-выражениями без захвата