Языки программирования — вопросы к промежуточному экзамену

Вопросы на итоговом экзамене относятся в первую очередь к языку C++, если не указано обратное. Указанные вопросы в билете в чистом могут не встречаться, но необходимы для решения задач в нём, а также при обсуждении их решения и дополнительных вопросов.

Экзамен проводится устно с подготовкой до 1 часа, на ответ можно взять только краткий план и запись ответов на вопросы, где это необходимо. В билете 3 вопроса, из которых первый — изложение среза теоретического материала, не более 5 минут. Ответ строить излагая основные определения и классификации, с формальной точки зрения с объяснением практической применимости. Два последних вопроса по формату соответствуют вопросам контрольной работы.

Вопросы оцениваются в 3 балла максимум. После ответа на них, если наличие дополнительного балла влияет на промежуточную оценку, задаётся дополнительный вопрос на этот балл.

1. Языки программирования. Классификации языков программирования (поколения, парадигмы, свойства системы типов). Языки C и C++.
2. Виды ошибок в программах: синтаксические, семантические, логические и ошибки времени выполнения.
3. Поведение программ на языке C++: определённое, неуточняемое, определяемое реализацией и неопределённое поведение.
4. Этапы трансляции программы на языке C++.
   - 4.1. Работа препроцессора. Директивы условной трансляции, макроподстановки и включения текстов.
   - 4.2. Лексический состав языка: токены и их виды.
5. Система типов языка C++.
   - 5.1. Фундаментальные типы: арифметические типы и их классификация, `void`, `std::nullptr_t`.
   - 5.2. Производные типы
     - 5.2.1. Указатели. Нулевые указатели.
     - 5.2.2. Леводопустимые ссылки.
     - 5.2.3. Массивы.
     - 5.2.4. Функции.
     - 5.2.5. Перечисления (с областью видимости и без).
     - 5.2.6. Классы. Статические и нестатические члены классов. Неявный параметр-объект и его адрес (`this`).
   - 5.3. Библиотечные типы: типы фиксированной ширины, `std::size_t`, `std::ptrdiff_t`.
6. Приведения типов.
   - 6.1. Явные и неявные преобразования.
   - 6.2. Преобразования между арифметическими типами.
   - 6.3. Разложения lvalue в rvalue и массивов в указатели. Материализация временных значений.
   - 6.4. Преобразования квалификаторов и указателей.
   - 6.5. Пользовательские преобразования: конструкторы и функции преобразования, явные преобразования классов (`explicit`).
   - 6.6. Приведение типов в функциональном стиле. Временные объекты.
7. Выражения. Вычисление выражений. Категории значений. Константные выражения и спецификатор `constexpr`.

8. Литералы: целочисленные, с плавающей точкой, символьные, булевы. Литерал указателя. Строковые литералы. Пользовательские литералы.
9. Объекты и доступ к ним (чтение/запись). Изменяемость объектов, квалификатор `const`. Выравнивание объектов.
10. Операции.
    - 10.1. Операции приведения типов `static_cast` и `const_cast`.
    - 10.2. Арифметические операции. Целочисленные повышения, повышения с плавающей точкой и обычные арифметические преобразования.
    - 10.3. Логические операции. Вычисления по короткой схеме.
    - 10.4. Побитовые операции.
    - 10.5. Условная операция.
    - 10.6. Операции отношения и равенства.
    - 10.7. Простое и составные присваивания, инкремент и декремент.
    - 10.8. Операции `sizeof` и `alignof`.
    - 10.9. Операции взятия адреса и разыменования.
    - 10.10. Арифметика указателей.
    - 10.11. Операций вызова функции. Рекурсия.
    - 10.12. Сокращения: косвенная выборка и индексация.
    - 10.13. Операция «запятая».
    - 10.14. Перегрузка операций.
11. Описания и определения. Правило одного определения.
    - 11.1. Описания стандартной формы.
        - 11.1.1. Определения функций.
        - 11.1.2. Аргументы по умолчанию.
    - 11.2. Псевдонимы типов.
    - 11.3. Описание `using`.
    - 11.4. Описания пространств имён и их псевдонимов.
12. Инициализация.
    - 12.1. Инициализация по умолчанию.
    - 12.2. Инициализация нулём, константная инициализация, статическая и динамическая инициализация объектов со статическим временем хранения.
    - 12.3. Инициализация копированием в описаниях и на границах функций.
    - 12.4. Прямая инициализация в описаниях, приведениях типов и конструкторах.
    - 12.5. Инициализация значения.
    - 12.6. Инициализация списком. Универсальная инициализация. Сужающие преобразования.
    - 12.7. Инициализация классов: конструкторы. Инициализаторы нестатических членов данных. Делегирование конструкторов.
    - 12.8. Привязка ссылок. Прямая и непрямая привязки.
13. Области видимости: блок, функция (для меток), параметр функции (для параметров функции), пространство имён, класс, перечисление, параметр шаблона.
14. Связанность: внутренняя, внешняя и отсутствующая. Анонимные пространства имён и связанность. Встраиваемые сущности (`inline`) и слабая связанность.
15. Время хранения: автоматическое и статическое. Временные объекты и их привязка к ссылкам.
16. Влияние спецификаторов `static` и `extern` на характеристики описания.
17. Поиск имён.

17.1. Неквалифицированные имена и их поиск в блоках, пространствах имён и классах. Скрытие имён.

17.2. Аргументо-зависимый поиск имён.

17.3. Квалифицированные имена и их поиск.

17.4. Директивы `using` и их влияние на поиск имён.

17.5. Разрешение перегрузок.

17.6. Подобъекты неявного параметра-объекта в поиске имён.

17.7. Поиск имён во вложенных и локальных классах.

17.8. Уровни доступа членов класса, друзья классов.

17.9. Шаблоны функций в разрешении перегрузок и их частичный порядок.

18. Шаблоны.

18.1. Инстанциация и специализации.

18.2. Явная и частичная специализация шаблонов классов.

18.3. Описания и определения явной инстанциации.

19. Операторы.

19.1. Оператор-выражение. Пустой оператор.

19.2. Блок.

19.3. Ветвления: `if` (+`else`), `switch`.

19.4. Циклы: `while`, `do...while`, `for`.

19.5. Описания в блоках и контролирующих операторах.

19.6. Операторы перехода: `break`, `continue`, `return`, `goto`. Помеченный оператор.

20. Атрибуты. `[[fallthrough]]`. `[[noreturn]]`.

21. Параметры функций. Связь входных/выходных параметров с указателями, массивами и ссылками. `const`-корректность.

22. Обеспечение инвариантов класса: инкапсуляция. Свойства и акцессоры.

23. Интерфейсы единиц трансляции. Заголовочные файлы и их защита от повторного включения.

24. Функционирование скомпилированных программ (в общем и на примере x86-64)

24.1. Устройство центрального процессора и памяти в рассматриваемой архитектуре.

24.2. Представление в памяти всех типов данных.

24.3. Структура простых команд на языке ассемблера, трансляция выражений.

24.4. Конструкции структурного программирования в машинном коде.

24.5. Вызов процедур в машинном коде. Соглашения о вызовах. Аппаратный стек и кадры стека. Реализация автоматического времени хранения.

24.6. Устройство объектных файлов. Секции и символы. Реализация внешней и слабой связанностей и статического времени хранения.

25. Инструментальные средства. gcc-совместимые драйверы компиляторов. Система сборки CMake. Интегрированные среды разработки. Интерактивные отладчики. Средства статического анализа. Средства динамического анализа на примере sanitizer'ов. Анализ покрытия.

26. Тестирование ПО. Предусловия, постусловия и их проверки утверждениями.

Языки программирования. Промежуточный экзамен. Билет № 1.

1. Виды ошибок в программах. Виды определённости поведения в языке C++. Инструментальные средства для обнаружения ошибок.
2. Объясните полностью смысл следующего описания:
   ```
   const int*& x{static_cast<int*>(nullptr)};
   ```
3. Предложите возможные интерпретации значения, имеющего представление `81 04` в шестнадцатеричном дампе, если всё, что о нём известно - что оно целочисленное.

Языки программирования. Промежуточный экзамен. Билет № 2.

1. Возможности этапа предварительной обработки и их альтернативы в современном C++.
2. Укажите размер объекта типа
   ```
   struct { void (*a[2][5])(); char b; unsigned short c; } [3]
   ```
   на архитектуре x86-64.
3. Декомпилируйте в возможное определение на языке C++ функцию, которой соответствует фрагмент ассемблера
   ```
   mov rax,rsp
   cmp rax,0
   jns .a ; jump if not sign
   neg rax
   .a: ret
   ```

Языки программирования. Промежуточный экзамен. Билет № 3.

1. Этапы трансляции программы на языке C++. Устройство объектных файлов и образов программ.
2. Прокомментируйте с точки зрения const-корректности описания функции `strstr` в стандартных библиотеках языков C и C++.
3. Приведите примеры трансляции на язык ассемблера x86 всех операторов перехода.

Языки программирования. Промежуточный экзамен. Билет № 4.

1. Лексический состав языка C++. Виды литералов.
2. Укажите вхождения неполных типов в конструкциях фрагмента
   ```
   void f(int x[]) { extern int y[]; struct S {}; int z[] = {0}; }
   ```
3. Укажите ожидаемые вами записи таблицы символов объектного файла, соответствующего следующей единице трансляции, с указанием для каждого символа, известен был его адрес компилятору или нет:
   ```
   #include <iostream>
   static int e = 1;
   void f() { extern int a; std::cout << a << '\n'; }
   ```

Языки программирования. Промежуточный экзамен. Билет № 5.

1. Фундаментальные типы языка C++.
2. Разберите процесс инициализации в определении
   ```
   double x[][3] = {{alignof(char)},{{},1ULL*1.f}};
   ```
3. Объясните, каким образом интерактивный отладчик получает информацию о цепочке функций, выполняемых в приостановленной программе.

Языки программирования. Промежуточный экзамен. Билет № 6.

1. Константные выражения и контексты их применения.
2. Докажите, что язык C++ поддерживает структурную и процедурную парадигмы программирования.
3. Как реализуются пространства имён, метки и преобразования квалификаторов в машинном коде?

Языки программирования. Промежуточный экзамен. Билет № 7.

1. Указатели. Массивы и арифметика указателей.
2. В каких случаях пробелы в программе на языке C++ являются значащими?
3. Приведите примеры определений, которые будут расположены во всех известных вам секциях объектного файла.

Языки программирования. Промежуточный экзамен. Билет № 8.

1. Ссылки и их инициализация.
2. Объясните процесс разрешения перегрузок

   ```
   void f(int* x,int y);
   void f(int x[10],const int y);
   void f(std::nullptr_t np,double z);
   ```

   для вызова `f(0,'a')`.
3. Может ли значение 1.2 быть в точности представлено форматом с двойной точностью по стандарту ISO/IEC/IEEE 60559:2011?

Языки программирования. Промежуточный экзамен. Билет № 9.

1. Функции. Операция вызова функции.
2. Разберите процесс вычисления выражения

   ```
   *("123"+2)+alignof(short)
   ```

   в рамках архитектуры x86-64.
3. Каков порядок использования встроенных (intrinsic) функций компилятора?

Языки программирования. Промежуточный экзамен. Билет № 10.

1. Неизменяемые объекты. Преобразования квалификаторов. const-корректность.
2. Укажите виды инициализации, использованные в фрагменте
   ```
   int f(int x) { int y{}; return y; }
   ```
3. Почему замена арифметических операций на эквивалентные выражения с использованием побитовых может принести улучшения?

Языки программирования. Промежуточный экзамен. Билет № 11.

1. Метки и безусловные переходы. Целесообразность использования и эквиваленты структурного программирования.
2. Разберите процесс вычисления выражения
   ```
   std::cout << +'7' << (0,1,2)
   ```
3. Предложите реализацию на ассемблере функции
   ```
   int f(int (*p)[3]){ return p[2][1]; }
   ```

Языки программирования. Промежуточный экзамен. Билет № 12.

1. Типы-перечисления.
2. Разберите процесс вычисления выражения
   ```
   !const_cast<const char*>(static_cast<char*>(nullptr))+0.
   ```
3. Может ли `00 FF 00 FF 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF` являться представлением классового типа, являющегося структурой данных с переменным во время работы программы числом элементов?

Языки программирования. Промежуточный экзамен. Билет № 13.

1. Доступ к отдельным битам и группам битов целочисленных значений.
2. Сравните различные способы создания функций с несколькими выходными параметрами.
3. Объясните механизм определения параметров шаблона в следующей ситуации:

```
template<int X = 42,typename T = double,typename U = int,std::size_t N = 0>
int f(T (&)[N]);
int b[7],a = f<5>(b);
```

Языки программирования. Промежуточный экзамен. Билет № 14.

1. Классы. Статические и нестатические члены классов и доступ к ним. Неявный параметр-объект.
2. Какая имеется альтернатива аргументам функций по умолчанию?
3. Объясните процесс разрешения перегрузок на примере:

```
int f(int*,double);
template<typename T> int f(T*,T);
template<typename T,typename U = double> double f(T*,U);
int x = f<int>(nullptr,3.5);
```

Языки программирования. Промежуточный экзамен. Билет № 15.

1. Преобразования, затрагивающие категорию значения и из роль в языке.
2. Какие средства позволяют устранить дублирования кода в перегрузках конструкторов?
3. Предложите улучшения для функции

```
void f(int* p,int n) { for(int i=0;i<n;++i) std::cout << p[i] << '\n'; }
```

Языки программирования. Промежуточный экзамен. Билет № 16.

1. Приведения между арифметическими типами. Сужающие преобразования.
2. Разберите на токены строку
   ```
   int abc = def+++"\t"[12_cm];
   ```
3. Объясните назначение статического, динамического и анализа покрытия.

Языки программирования. Промежуточный экзамен. Билет № 17.

1. Пользовательские преобразования.
2. Почему следует предпочитать инициализацию присваиванию? Как это проявляется для классовых типов?
3. Укажите представление в памяти архитектуры x86-64 типа
   ```
   struct S { constexpr static unsigned x = -1; const S& s; inline void f()  }
   ```

Языки программирования. Промежуточный экзамен. Билет № 18.

1. Определение результирующего типа операций над арифметическими типами.
2. В чём различие перегрузки операций в виде членов класса и свободных функций?
3. В каких случаях в прологе функции на архитектуре x86-64 можно избежать изменения регистра `rsp`?

Языки программирования. Промежуточный экзамен. Билет № 19.

1. Операции с побочными эффектами.
2. Для чего могут применяться описания и определения явной инстанциации шаблонов?
3. Назовите причину остановки машинного кода программы, соответствующего единице трансляции

```
void f() { f(); }
int main() { f(); }
```

Языки программирования. Промежуточный экзамен. Билет № 20.

1. Характеристики сущностей в описаниях стандартной формы.
2. Укажите контексты применения списоков инициализации и их роль в них.
3. Приведите примеры корректного использования отрицательных и вещественных значений в качестве операндов операции индексации.

Языки программирования. Промежуточный экзамен. Билет № 21.

1. Псевдонимы типов и их использование в стандартной библиотеке.
2. Разберите процесс вычисления выражения

```
-static_cast<short>("4780"[1]-"256"[2]);
```

3. Останется ли корректной произвольная программа после удаления из неё всех ключевых слов `inline`?

Языки программирования. Промежуточный экзамен. Билет № 22.

1. Времена хранения и их реализация в машинном коде.
2. Перечислите операции, операнды которых могут не вычисляться и укажите, при каких условиях.
3. Объясните проблему в единице трансляции

```
constexpr int f(int x) { return x*2; }
int main() { const double d = 3.5; int a[f(d)]; }
```

Языки программирования. Промежуточный экзамен. Билет № 23.

1. Поиск неквалифицированных имён (без разрешения перегрузок и аргументо-зависимого поиска).
2. Укажите следующие атрибуты всех упомянутых ниже имён (если применимо): область видимости, связанность, время хранения, является ли это их описанием или определением:

```
static int x;
struct s { int a; } y;
void f(bool b) { extern double g; }
```

3. Найдите ошибки в функции

```
void zero(int x[]) { for(unsigned i=0;i<sizeof(x);i++) x[i] = 0; }
```

Языки программирования. Промежуточный экзамен. Билет № 24.

1. Поиск квалифицированных имён. Описания и директивы `using`.
2. Опишите процесс поиска имени `x` в последней строке фрагмента:

```
struct S { static int x; void f(); };
int S::x;
int x;
void S::f() { x; // <-- }
```

3. Укажите элементы языка, для реализации которых в настоящее время применяются расширения классической модели компоновки только с двумя видами видимости символов — глобальной и локальной.

Языки программирования. Промежуточный экзамен. Билет № 25.

1. Разрешение перегрузок.
2. Укажите, сколько максимально объектов, идентифицируемых `x` в любой области видимости, может одновременно существовать в следующей программе и почему:

```
int x = 3;
void f(int x) { int y = x; static int x; if(y) f(y-1); }
int main() { f(x); }
```

3. Сколько вызовов и каких инструментальных средств произведёт система сборки с реализацией процесса напрямую в соответствии со стандартом языка, чтобы собрать с нуля программу из 7 файлов исходного текста и 10 заголовочных файлов?

Языки программирования. Промежуточный экзамен. Билет № 26.

1. Перегрузка операций. Аргументо-зависимый поиск имён.
2. Запишите с использованием только операторов ветвления и безусловного перехода конструкции, аналогичные циклу с предусловием, циклу с постусловием и циклу `for`.
3. Может ли таблица перемещения объектного файла содержать записи, соответствующие объектам с автоматическим временем хранения?

Языки программирования. Промежуточный экзамен. Билет № 27.

1. Инкапсуляция и её реализация в C++. Свойства и функции доступа (акцессоры).
2. Запишите с использованием только цикла с предусловием `while` и составного операторов конструкции, аналогичные оператору ветвления с обоими ветвями, циклу с постусловием и циклу `for`. Допускается конечное дублирование кода и введение новых объектов.
3. Опишите процесс поиска имени `f` в последней строке фрагмента

```
namespace A {
struct Z {};
void f(const Z&);
}
int main() {
f(A::Z{}); }
```

Языки программирования. Промежуточный экзамен. Билет № 28.

1. Шаблоны функций и классов и задание их особых форм для более конкретных наборов значений параметров.
2. Опишите процесс поиска имени `x` в последней строке фрагмента

```
namespace A{
int x;
struct S { void f(); };
}
int x;
void A::S::f() { x; // <-- }
```

3. Объясните назначение драйвера компилятора при наличии компилятора и системы сборки при наличии драйвера компилятора.

Языки программирования. Промежуточный экзамен. Билет № 29.

1. Интерфейсы единиц трансляции и использование в них различных конструкций языка.
2. Приведите пример возникновения висячей ссылки, которая была привязана к объекту с автоматическим временем хранения.
3. Объясните, почему аппаратный стек называют «аппаратным»?

Языки программирования. Промежуточный экзамен. Билет № 30.

1. Использование утверждений в программах.
2. Запишите определение леводопустимой ссылки со статическим временем хранения в блочной области видимости, привязанной к объекту типа указатель на массив из 10 значений типа `int`, расположенного в памяти непосредственно перед таким же значением, идентифицируемым возвращаемым значением функции `f`, вызванной с константой нулевого указателя в качестве аргумента.
3. Объясните различие между физическими и виртуальными адресами.