**Факультет инфокоммуникационных систем и технологий**

**КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ**

**Ответ на билет № 3**

**по дисциплине «Технологии и методы программирования»**

**Выполнил:** студент гр. ИБО-ЗИ-20, Шайдулин С. В.

**Преподаватель:** Старший преподаватель, Строганова С. М.

**Королёв, 2022 г.**

**Билет № 3**

**1. Операции над скалярными данными С++. Приоритеты операций. Примеры выражений.**

**2. Инициализация полей при отсутствии конструктора. Пример.**

**1. Для начала необходимо определить, что такое «скалярные данные».**

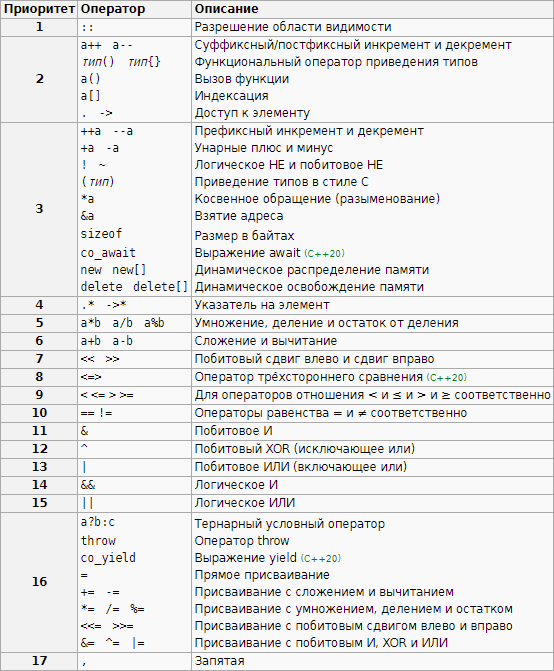
Все допустимые в языке ТП типы подразделяются на две большие группы: скалярные и структурированные. К стандартным скалярным типам относятся данные **целочисленного**, **вещественного**, **литерного** и **булевского** типов.

Скалярный тип: тип, содержащий одно значение определенного диапазона. Скаляры включают арифметические типы (целочисленные или значения с плавающей запятой), элементы типа перечисления, типы указателей, типы указателей на члены и std::nullptr\_t. Основными типами обычно являются скалярные типы.

Последовательность выполнения операций в выражении определяется тремя факторами:

* приоритетом операций;
* порядком расположения операции;
* использованием скобок.

Приоритеты выполняемых операций можно выразить следующей таблицей:

  
Рисунок 1. Таблица приоритетов операторов.

В качестве примера возьмём следующее выражение:

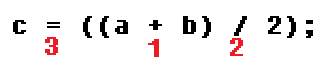


Рисунок 2. Простое выражение.

Стоит сказать, что оператор присваивания начинает читать строку слева направо, до тех пор, пока не встретит «;». Пока не встречен этот знак, приоритет действий выставляется либо скобками, либо приоритетом операторов.

**2. Инициализация полей при отсутствии конструктора. Пример.**

В языке C++ есть много способов инициализировать переменные и поля. Как минимум, существует 19 способов инициализировать простой int. Инициализировать поля можно как с помощью конструктора, как и без него.

Типовой пример:

|  |  |
| --- | --- |
| В первом случае вызываются конструкторы для полей, и тем самым осуществляется инициализация полей. | Во втором случае сначала вызываются конструкторы по умолчанию, и, если инициализаторы полей отсутствуют в их определении, то затем еще вызываются копирующие операторы присваивания, чтобы инициализировать соответствующим образом эти поля, что может быть в конечном итоге очень затратно. |
| MyClass(): a(1),b(2),c(3)  { } | MyClass()  {  a = 1;  b = 2;  c = 3;  } |

Рисунок 3. Типовой пример.

В качестве дополнительного примера, можно привести такой следующий код:

|  |
| --- |
| class СBook{  public: char Name[30];  int Pages;  char \*getName() { // метод по умолчанию inline, так  return Name; // как его тело описано в классе  }  int getPages(); // тело будет описано в book.сpp  };  void main()  { СBook A = {"J.London. V.1",366};  СBook C[]= {{"J.London. V.3",367},  {"J.London. V.4",321},  {"J.London. V.5",356}};  ... } |

Рисунок 4. Типовой пример.

Если оба варианта рабочие, то практическая польза от инициализации вне тела конструктора заключается в порядке вызова. Это имеет смысл для классов, которые в качестве полей имеют объекты других классов. Таким способом можно избежать дубликат объекта класса, если при инициализации какого-нибудь его поля произойдёт ошибка. Ну и для объектов других классов могут быть затраты на инициализацию объекта. Поэтому может получиться повторный вызов. Ну и последний аргумент - вариант 1 является хорошей практикой.

Код, сгенерированный компилятором, инициализирует только те члены, у которых существуют конструкторы, поэтому при отсутствии конструктора будут неявно инициализированы только объектные члены классов. Члены, объявленные с встроенным типом, не инициализируются автоматически. Такое поведение логически согласовано с отсутствием инициализации структур в С.