**Доклад Никиты Луговского ШАД-212**

12 августа 2011 года ISO (Международная организация по стандартизации) одобрила новую версию C++, получившую название C++11. C++11 добавляет к языку C++ совершенно новый набор функциональных возможностей! Использование этого нового функционала совершенно необязательно, но вы, несомненно, найдете некоторое из него полезным. Все предыдущие руководства были обновлены, чтобы соответствовать C++11.

**Цели C++11**

Бьёрн Страуструп охарактеризовал цели C++11 так:

* Развивать сильные стороны C++ – вместо того, чтобы пытаться распространить C++ на новые области, где он может быть слабее (например, приложения Windows с тяжелым графическим интерфейсом), сосредоточиться на том, чтобы заставить его делать то, что у него хорошо получается, еще лучше.
* Сделать C++ более простым в изучении и использовании – предоставить функциональные возможности, которые делают язык более последовательным и простым в использовании.

С этой целью комитет, согласовавший стандарт C++11, попытался руководствоваться следующими общими принципами:

* По возможности сохранить стабильность и совместимость со старыми версиями C++ и C. Программы, работавшие на C++03, как правило, должны работать и на C++11.
* Свести количество расширений ядра языка к минимуму и поместить основную часть изменений в стандартную библиотеку (цель, которая в этом обновлении не была достигнута);
* Сосредоточиться на улучшении механизмов абстракции (классы, шаблоны), а не на добавлении механизмов для обработки конкретных ситуаций.
* Добавить новую функциональность как для новичков, так и для экспертов. Понемногу для всех!
* Повышение безопасности типов, чтобы предотвратить непреднамеренные ошибки.
* Повысить производительность и позволить C++ работать с оборудованием напрямую.
* Учитывать удобство использования и проблемы экосистемы. C++ должен хорошо работать с другими инструментами, быть простым в использовании и обучении и т.д.

C++11 не сильно отличается от C++03 в тематическом плане, но он добавил огромное количество нового функционала.

**Основные нововведения в C++11**

Вот список основных функций, добавленных в C++11. Обратите внимание, что этот список не является исчерпывающим, а скорее предназначен для выделения некоторых ключевых интересных особенностей.

* auto ([8.7 – Вывод типов для объектов с использованием ключевого слова auto](https://radioprog.ru/post/1132));
* char16\_t и char32\_t и новые литералы для их поддержки;
* constexpr ([4.14 – const, constexpr и символьные константы](https://radioprog.ru/post/1104));
* decltype;
* спецификатор default;
* делегирующие конструкторы ([12.8 – Перекрывающиеся и делегирующие конструкторы](https://radioprog.ru/post/1224));
* спецификатор delete ([13.14 – Конструктор преобразования, explicit и delete](https://radioprog.ru/post/1250));
* классы enum ([9.3 – Классы перечислений](https://radioprog.ru/post/1157));
* внешние шаблоны;
* лямбда-выражения ([11.13 – Введение в лямбды (анонимные функции)](https://radioprog.ru/post/1203)) и захваты ([11.14 – Лямбда-захваты](https://radioprog.ru/post/1204));
* long long int ([4.3 – Размеры объектов и оператор sizeof](https://radioprog.ru/post/1093));
* конструктор и присваивание перемещения ([M.3 – Конструкторы перемещения и присваивание перемещением](https://radioprog.ru/post/1307));
* спецификатор noexcept (краткое упоминание в [20.4 – Неперехваченные исключения и универсальные обработчики](https://radioprog.ru/post/1299));
* nullptr ([10.9 – Нулевые указатели](https://radioprog.ru/post/1169));
* спецификаторы override и final ([18.3 – Спецификаторы override и final, и ковариантные возвращаемые типы](https://radioprog.ru/post/1276));
* циклы for на основе диапазона ([10.19 – Циклы for-each (циклы на основе диапазона)](https://radioprog.ru/post/1180));
* ссылки на r-значения ([M.2 – Ссылки на r-значения (rvalue-ссылки)](https://radioprog.ru/post/1306));
* static\_assert ([7.17 – assert и static\_assert](https://radioprog.ru/post/1154));
* std::initializer\_list ([16.7 – Список инициализаторов std::initializer\_list](https://radioprog.ru/post/1262));
* синтаксис, завершающийся возвращаемым типом ([8.7 – Вывод типов для объектов с использованием ключевого слова auto](https://radioprog.ru/post/1132));
* псевдонимы типов ([8.6 – typedef и псевдонимы типов](https://radioprog.ru/post/1131));
* typedef теперь может определять тип шаблонных классов;
* унифицированная инициализация ([4.1 – Введение в основные типы данных](https://radioprog.ru/post/1091));
* определяемые пользователем литералы;
* вариативные шаблоны;
* >> теперь будет правильно интерпретироваться как закрытие объекта шаблона.

В стандартной библиотеке C++ также появилось много новых классов, доступных для использования.

* лучшая поддержка многопоточности и локального хранилища потоков;
* хеш-таблицы;
* улучшения генерирования случайных чисел (основное обсуждение в [9.5 – Генерирование случайных чисел](https://radioprog.ru/post/1159));
* обертки ссылок ([18.9 – Нарезка объектов](https://radioprog.ru/post/1282));
* регулярные выражения;
* std::auto\_ptr устарел ([M.1 – Введение в умные указатели и семантику перемещения)](https://radioprog.ru/post/1236);
* std::tuple (краткое упоминание в [11.5 – Возвращение значений по значению, по ссылке и по адресу](https://radioprog.ru/post/1192));
* std::unique\_ptr ([M.6 – std::unique\_ptr](https://radioprog.ru/post/1310)).