Introduction to C++0x

Alexander Færøy

Open Source Community Day 2009

October 24, 2009

Outline

Example without C++0x

```
class Person {
    private:
        string _name;
        bool _child;
    public:
        Person(const string & n) : _name(n), _child(false) {
        }

        Person(const string & n, bool c) : _name(n),
            _child(c) {
        }

        Person(const Person & other) {
            _name = other._name;
        }

        Person & operator= (const Person & other) {
            _name = other._name;
            return *this;
        }

        string get_name() const { return _name; }
        bool is_adult() const { return ! _child; }
};
```

Example without C++0x

```
int main(int, char **) {
    vector<Person> persons;

    persons.push_back(Person("A"));
    persons.push_back(Person("B"));

    for (vector<Person>::iterator t(persons.begin()),
            t_end(persons.end()); t != t_end; ++t) {
        string name(t->get_name());
        bool adult(t->is_adult());

        cout << "Name: " << name << " is "
            << adult ? "an adult" : "a child" << endl;
    }

    return EXIT_SUCCESS;
}
```

Type Inference

N1984

```
int main(int, char **) {
    vector<Person> persons;

    persons.push_back(Person("A"));
    persons.push_back(Person("B"));

    for (auto t(persons.begin()), t_end(persons.end());
            t != t_end; ++t) {
        auto name(t->get_name());
        auto adult(t->is_adult());

        cout << "Name: " << name << " is "
            << adult ? "an adult" : "a child" << endl;
    }

    return EXIT_SUCCESS;
}
```

Initializer Lists

N2672

```
int main(int, char **) {
    vector<Person> persons = { Person("A"), Person("B") };

    for (auto t(persons.begin()), t_end(persons.end());
            t != t_end; ++t) {
        auto name(t->get_name());
        auto adult(t->is_adult());

        cout << "Name: " << name << " is "
            << adult ? "an adult" : "a child" << endl;
    }

    return EXIT_SUCCESS;
}
```

Default and Deleted Functions

N2346

```
class Person {
    private:
        string _name;
        bool _child;
    public:
        Person(const string & n) : _name(n), _child(false) {
        }

        Person(const string & n, bool c) : _name(n),
            _child(c) {
        }

        Person(const Person &) = default;
        Person & operator= (const Person &) = default;

        string get_name() const { return _name; }
        bool is_adult() const { return ! _child; }
};
```

Delegated Constructors

N1986

```
class Person {
    private:
        string _name;
        bool _child;
    public:
        Person(const string & n) :
            Person(n, false) {
        }

        Person(const string & n, bool c) : _name(n),
            _child(c) {
        }

        Person(const Person &) = default;
        Person & operator= (const Person &) = default;

        string get_name() const { return _name; }
        bool is_adult() const { return ! _child; }
};
```

Range-based For-loops

N2930

```
int main(int, char **) {
    vector<Person> persons = { Person("A"), Person("B") };

    for (auto t : persons) {
        auto name(t.get_name());
        auto adult(t.is_adult());

        cout << "Name: " << name << " is "
            << adult ? "an adult" : "a child" << endl;
    }

    return EXIT_SUCCESS;
}
```

Questions

Any Questions?