Программирование на языке C++
Лекция 6

Специализация шаблонов

Александр Смаль

Полная специализация шаблонов: классы

```
template<class T>
struct Array {
    ...
    T *   data_;
};
template<>
struct Array<bool> {
    static int const INTBITS = 8 * sizeof(int);
    explicit Array(size_t size)
        : size_(size)
        , data_(new int[size_ / INTBITS + 1])
    {}
    bool operator[](size_t i) const {
        return data_[i / INTBITS] & (1 << (i % INTBITS));
    }
private:
    size_t  size_;
    int *  data_;
};
```

Полная специализация шаблонов: функции

```
template<class T>
void swap(T & a, T & b)
{
    T tmp(a);
    a = b;
    b = tmp;
}

template<>
void swap<Database>(Database & a, Database & b)
{
    a.swap(b);
}

template<class T>
void swap(Array<T> & a, Array<T> & b)
{
    a.swap(b);
}
```

Специализация шаблонов функций и перегрузка

```
template<class T>
void foo(T a, T b) {
    cout << "same types" << endl;
}

template<class T, class V>
void foo(T a, V b) {
    cout << "different types" << endl;
}

template<>
void foo<int, int>(int a, int b) {
    cout << "both parameters are int" << endl;
}

int main() {
    foo(3, 4);
    return 0;
}
```

Частичная специализация шаблонов

```
template<class T>
struct Array {
    T & operator[](size_t i) { return data_[i]; }
    ...
};

template<class T>
struct Array<T *> {
    explicit Array(size_t size)
        : size_(size)
        , data_(new T *[size_])
    {}

    T & operator[](size_t i) { return *data_[i]; }

private:
    size_t  size_;
    T **    data_;
};
```