Метапрограммная реализация библиотеки арифметики в простых конечных полях и их расширениях

А. С. Мышко

Направление подготовки: Прикладная математика и информатика
Руководитель: асс. каф. ИВЭ А. М. Пеленицын

Южный федеральный университет
Институт математики, механики и компьютерных наук имени И. И. Воровича

Кафедра информатики и вычислительного эксперимента

Постановка задачи

1. Реализовать арифметику в конечных полях и их расширениях с использованием метапрограммирования на шаблонах C++.
2. Исследовать возможности стандарта C++11 и преимущества их применения.

Черты функционального программирования:

- чистые функции;
- рекурсия;
- функции высших порядков.

Пример: возведение числа 3 в степень N

```
template<int N>
struct Pow3 {
        enum { result = 3 * Pow3<N-1>::result };
        typedef int_<result> type;
};

template<>
struct Pow3<0> {
        enum { result = 1 };
        typedef int_<result> type;
};
```

Простые поля

Шаблон для элемента простого поля

```
template <int N, int x>
struct FFE {
    enum { value = x % N };
    typedef int_<value> type;
};
```

Статические константы:

- Объявление внутри шаблона: **enum** / **static const**;
- Обёртки: Boost.MPL, C++11 (<type_traits>).

Простые поля

Сложение двух элементов поля

```
int main(){
    typedef FFE<7, 5> a;
    typedef FFE<7, 4> b;
    return FFE_sum<7, a, b>::type::value;
}
```

Ассемблерный листинг (gcc 4.9.0, -S -O0 -std=c++11)

```
    pushq   %rbp
    movq    %rsp, %rbp
    movl    $2, %eax
    popq    %rbp
    ret
```

Проверка на простоту

Проверка на простоту с использованием constexpr

```
constexpr
bool is_prime_recursive(size_t number, size_t c){
  return (c*c > number) ? true :
            (number % c == 0) ? false :
               is_prime_recursive(number, c+1);
}

constexpr bool is_prime_func(size_t number){
  return (number <= 1) ? false : is_prime_recursive
     (number, 2);
}
```

Проверка на простоту

Проверка на простоту на разных этапах

```
int main(){
    // compile-time
    static_assert(is_prime_func(7), "...");

    // run-time
    int i = 11;
    int j = is_prime_func(i));
}
```

Многочлены

Шаблон для многочлена над простым полем с использованием variadic templates

```
template <int N, int ... Nums>
struct Poly {
    static FFE_x<N> elements[sizeof ...(Nums)];
    enum { elements_count = sizeof ...(Nums) };
};

template<int N, int... Nums>
FFE_x<N> Poly<N, Nums... >::elements[] = {Nums ...};
```

Полученные результаты

1. Разработана метапрограммная библиотека, реализующая арифметику в простых полях и многочленах над ними.
2. Исследованы и применены следующие возможности стандарта C++11:
   - спецификатор `constexpr`;
   - объявление `static_assert`;
   - использование шаблонов с переменным количеством параметров (variadic templates).