C++0x: Конвертируем лямбда-выражение в указатель на функцию из песочницы

C++*

По роду деятельности мне часто приходится иметь дело с вычислительными задачами. В них нередко нужно передавать указатель на функцию, чтобы, например, построить график этой функции, или решить уравнение. Кроме того, указатели на функцию обычно используются в различных GUI фреймворках, чтобы указать, какое действие будет совершено при нажатии на определённую кнопку.

В новом стандарте C++0x появились зымыкания. Не вдаваясь в подробности, замыкания — это такие объекты, которые позволяют создавать функции прямо в теле других функций. Если подробнее — замыкания позволяют создавать функциональные объекты — то есть объекты, для которых определён **operator()**. На хабре уже писали о них: например тут.

Мне очень понравилось нововведение и я начал им пользоваться. Но только вот незадача: по смыслу, замыкания и функции — почти одно и то же, а использовать замыкания там, где должны использоваться указатели на функции, сходу не получается. По стандарту, замыкания без списка захвата должны свободно конвертироваться в указатели на функции, но на практике такого не наблюдалось, видимо ещё не реализовано. И я задался вопросом, можно ли использовать замыкания там, где используются указатели на функции?

Рассмотрим пример. Пусть у нас уже есть функция **printFunctionTable**, которая позволяет распечатать таблицу значений функции, при этом аргумент пробегает значения от 1 до 10.

```
void printFunctionTable(int (*func)(int)){
for(int i=1;i<=10;i++) cout << func(i) << " ";
cout << endl;
}
```

Конечно мы могли бы переписать эту функцию сразу под замыкания, а не под указатели на функции, но такая функция может быть расположена в закрытой библиотеке (как чаще всего и бывает), поэтому будем искать способ конвертации одного в другое.

Так же определим функцию, значения которой нам нужно распечатать:

```
int square(int x){
return x*x;
}
```

Теперь напишем главную функцию, в которой выведем таблицу квадратов функций.

```
int main(void)
{
printFunctionTable(square);
```

```
return 0;
}
```

Компилируем и запускаем, получаем табличку:

```
1 4 9 16 25 36 49 64 81 100
```

Теперь попробуем провернуть то же самое с замыканием — кубом числа.

```
auto cube=[](int x){
return x*x*x;
}

printFunctionTable(cube);
```

Не выходит — при компиляции — ошибка! Конечно — типы то — разные!
Как же нам это обойти? На помощь нам придут шаблоны.

```
template <typename CL>
int closureToFunction(int x){
CL cl;
return cl(x);
}
```

Тут мы определили шаблон функции, параметризованный типом **CL** — то есть типом нашего замыкания. Всё что нам нужно от нашего замыкания — вызвать его и вернуть значение.

Теперь можно использовать замыкание в том месте, где мы раньше использовали указатель на функцию. Выглядеть это будет так:

```
printFunctionTable(closureToFunction<decltype(cube)>);
```

после компиляции и запуска, получаем табличку:

```
1 8 27 64 125 216 343 512 729 1000
```

Урррa!!! — работает! =)

Для справки: **decltype** — это новое ключевое слово в C++0x, позволяющее определить тип выражения, то есть, в данном случае — тип нашего замыкания cube.

Вот собственно и всё — задача решена, теперь мы можем в удобном виде определять функции для каких-то действий прямо по ходу текста программы, а не скакать туда-сюда по исходнику, чтобы оформлять эти действия в отдельные функции.

Дополнительно хочется отметить, что мы создали только шаблон для сигнатуры int(int). По мере надобности можно добавлять и другие сигнатуры — double(int,double) к примеру, итд. Также, для удобства можно сделать макрос

```
#define CLOSURE_TO_FUNCTION(cl) closureToFunction<decltype(cl)>
```

P.S. Тестировалось на компиляторе Intel C++ Compiler 11.1, по идее, должно работать в g++ 4.5 и в visual studio 2010, главное не забыть проставить при компиляции флаги, позволяющие использовать c++0x.

Полный код программы:

```
#include <iostream>

using namespace std;

void printFunctionTable(int (*func)(int)){
for(int i=1;i<=10;i++) cout << func(i) << " ";
cout << endl;
}

int square(int x){
return x*x;
}

template int closureToFunction(int x){
CL cl;
return cl(x);
}

#define CLOSURE_TO_FUNCTION(cl) closureToFunction <decltype(cl)>

int main(void)
{
printFunctionTable(square);

auto cube=[](int x){return x*x*x;};
auto quad=[](int x){return x*x*x*x;};

printFunctionTable(CLOSURE_TO_FUNCTION(cube));
printFunctionTable(CLOSURE_TO_FUNCTION(quad));

return 0;
}
```

c++, c++0x, замыкание, closure, lambda, decltype, указатель на функцию