

# λ функции и std::function

Калин Георгиев

8 април 2020 г.

## $\lambda$ функции

## Недостатъци на (\*)

```
template <class T>
void map (T a[], size_t n, T(*f)(T))
{
    for (size_t i = 0; i < n; i++)
    {
        a[i] = f(a[i]);
    }
}

int plus1 (int x)
{return x+1};
int plus2 (int x)
{return x+2};
int mult2 (int x)
{return x*2};

int main ()
{
    int a[] = {1,2,3};
    map (a,3,plus1);
    map (a,3,plus2);
    map (a,3,mult2);
}
```

## Недостатъци на (\*)

- необходимо е да имаме компилиран код, чиито адрес да вземем
- не можем да създаваме функции “в движение”, които да зависят от контекста

```
using doublefn = double (*) (double);

doublefn squared (doublefn f)
{
    return ???x->f(f(x))???
}

int main ()
{
    std::cout << squared(f)(3);
}
```

# Прости ламбда функции

```
map (a,3,[] (int x)->int{return x+1;});  
map (a,3,[] (int x)->int{return x+2;});  
map (a,3,[] (int x)->int{return x*2;});
```

`std::function`

# Извикваем (callable) обект

Всичко, над което може да се приложи оператор `()`.

- функции
- $\lambda$  функции
- структури/класове с оператор `()`
- `std::bind` изрази

## Извикваем (callable) обект

```
class increase
{
public:
    increase (int _inc):inc (_inc){}
    int operator () (int x)
    {
        return x+inc;
    }
private:
    int inc;
};

int main ()
{
    increase inc1(5), inc2(10);
    std::cout << inc1(1) << std::endl;
    std::cout << inc2(1) << std::endl;
}
```



## std::function

Обвива всякакви callable обекти.

```
std::function<int(int)> f;  
f = plus1;  
std::cout << f(1);  
  
f = increase(1);  
std::cout << f(1);  
  
f = [](int x)->int{return x+1;};  
std::cout << f(1);
```

# Capture list

```
using doublefn = std::function<double(double)>;

doublefn squared (doublefn f)
{
    return [f](double x)
        ->double{return f(f(x));};
}

int main ()
{
    std::cout << squared ([](double x)
        ->double{return x*x;}) (10);
}
```

# Примери

- $f(g(x))$
- $f_1(f_2(\dots f_k(x)\dots))$
- $f^k(x)$
- Функция по списък от двойки

`std::bind` изрази

## Фиксиране на аргументи на функция

```
double quadratic (double a, double b, double c, double x)
{return a*x*x + b*x + c;}
```

```
int main ()
{
    using namespace std::placeholders;
    std::function<double(double)>
        qf = std::bind(quadratic,4,2,1,_1);

    std::cout << qf (0);
}
```

# Метод като функция

```
struct power
{
    double operator () (double x)
    {return pow (x, exp);}
    double exp;
};

int main ()
{
    using namespace std::placeholders;
    power pw2 = power{2};
    std::function<double(double)>
        squared = std::bind (&power::operator(), &pw2, _1);

    std::cout << squared(4);
}
```

# Алтернативен подход с $\lambda$

```
double quadratic (double a, double b, double c, double x)
{return a*x*x + b*x + c;}

struct power
{
    double operator () (double x)
    {return pow (x,exp);}
    double exp;
};

int main ()
{
    std::function<double(double)>
        qf2 = [](double x)->double{return quadratic(4,2,1,x);};

    power pw2 = power{2};

    std::function<double(double)>
        squared2 = [&pw2](double x)->double{return pw2(x);};

    std::cout << qf2 (0) << std::endl;
    std::cout << squared (4) << std::endl;
}
```

Въпроси?