

Федеральное государственное бюджетное образовательное учреждение высшего образования
Национальный исследовательский университет «Московский энергетический институт»
Кафедра ВМСС

Лабораторная работа №7

«Функциональное программирование в C++»

Курс: Технологии разработки программного обеспечения

Группа: А-07м-23

Выполнил: Кретов Н.В.

Проверила: Раскатова М.В.

Москва 2024

СОДЕРЖАНИЕ

1. Цель.....	3
2. Вариант задания	3
3. Разработка	4
4. Результат работы программы	5
ВЫВОД.....	6
ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММЫ.....	7

1. Цель

Главной задачей данной лабораторной работы является закрепление навыков функционального программирования на языке C++.

2. Вариант задания

Реализовать на языке C++ три задания по варианту (см. Таблицу 1), применяя принципы функционального программирования.

Язык C++, в отличие от Lisp, Erlang, SML, F# или Scala, не является функциональным, однако некоторые признаки данной парадигмы он содержит: можно создать функтор, перегрузив оператор круглых скобок, использовать анонимные (лямбда) функции, а также использовать указатели на функции.

В данной работе было применено использование лямбда-функций.

Таблица 1

Вариант задания к лабораторной работе №7

Вариант	Требуемые функции
14	2. Функция, возвращающая новый список из значений функции-аргумента, примененной к элементам списка-аргумента
	3. Функция, возвращающая значение, полученное путем применения функции-аргумента от двух значений к предыдущему результату полученному этой функцией и следующему элементу списка-аргумента.
	8. Функцию, генерирующую ассоциативный массив, где ключ – значение функции-аргумента от элемента списка-аргумента, а значение – количество таких значений

Так как в задании лабораторной работы не оговорено требование к наличию непосредственного взаимодействия с пользователем (по типу считывания значений, введенных в консоль), было принято решение не добавлять данный функционал, т.е. задача выполняется с заранее заданными параметрами, а пользователю виден лишь результат выполнения, выведенный в консоль.

3. Разработка

Для удобства разработки и поддержания структуры приложения было решено сопоставить каждой из заданных функций свой `#pragma region` (Task2, Task3 и Task8 – нумерации была выбрана в соответствии с нумерацией функций в задании к лабораторной работе).

В Task2 входят следующие функции:

- `Function2(int FunctionForCalculate(int), std::list<int> list)` – непосредственная реализация заданной функции 2 из Таблицы 1;
- `PrintList(std::list<int> list)` – вспомогательная функция для вывода списка, полученного в качестве параметра, в консоль;

В Task3 входит следующая функция:

- `Function3(int FunctionForCalculate(int, int), std::list<int> list)` – непосредственная реализация заданной функции 3 из Таблицы 1;

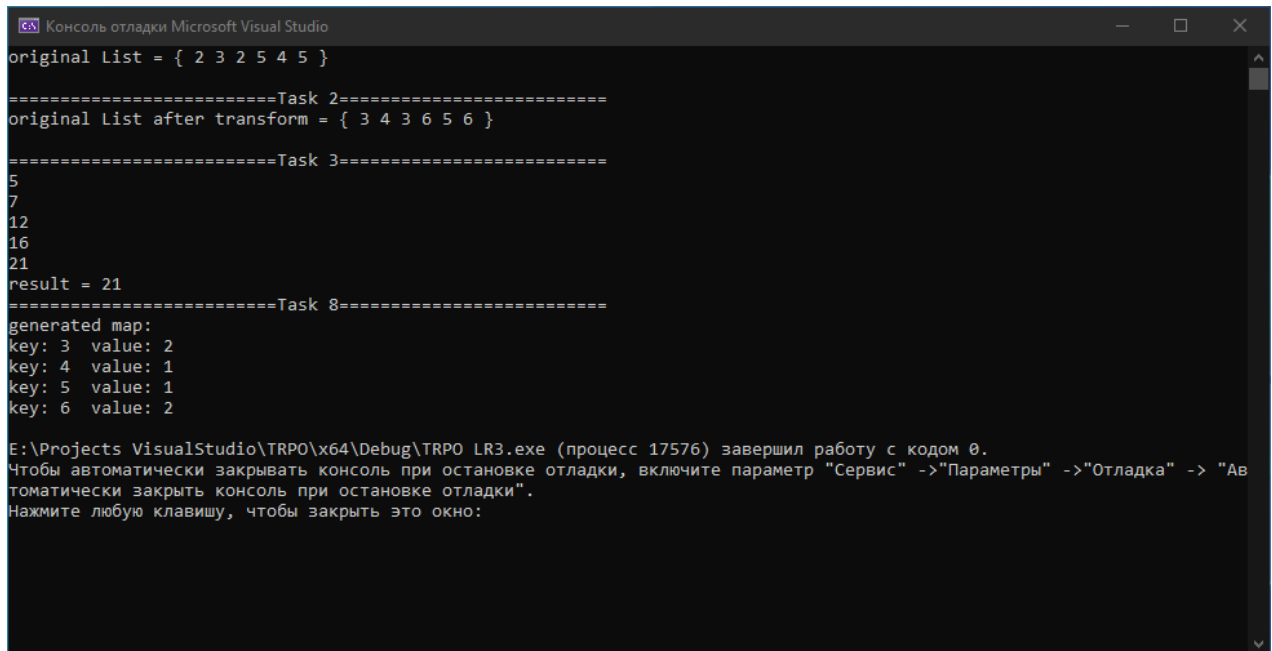
В Task8 входят следующие функции:

- `Function8(int FunctionForCalculate(int), std::list<int> list)` – непосредственная реализация заданной функции 8 из Таблицы 1;
- `PrintMap(std::map<int, int> map)` – вспомогательная функция для вывода ассоциативного массива, полученного в качестве параметра, в консоль;

С полным листингом программы можно ознакомиться в приложении А.

4. Результат работы программы

На рис. 1 представлен результаты работы программы.



```
Консоль отладки Microsoft Visual Studio
original List = { 2 3 2 5 4 5 }

=====Task 2=====
original List after transform = { 3 4 3 6 5 6 }

=====Task 3=====
5
7
12
16
21
result = 21

=====Task 8=====
generated map:
key: 3 value: 2
key: 4 value: 1
key: 5 value: 1
key: 6 value: 2

E:\Projects VisualStudio\TRPO\x64\Debug\TRPO LR3.exe (процесс 17576) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рис. 1. Результат работы программы

Кратко опишем представленные результаты:

В первой строке нам видно содержание программно заданного списка (2, 3, 2, 5, 4, 5).

После вывода заголовка Task2 показан результат вызова функции `Function2(int FunctionForCalculate(int), std::list<int> list)`. В качестве функции-аргумента была использована лямбда-функция `[](int number) {return ++number; }()`. В итоге к каждому элементу изначального списка была применена функция инкремента и результирующий список был выведен в консоль.

После вывода заголовка Task3 показан результат вызова функции `Function3(int FunctionForCalculate(int, int), std::list<int> list)`. В качестве функции-аргумента была использована лямбда-функция `[](int number1, int number2) {return number1 + number2; }()`. В итоге была получена сумма элементов исходного списке (последовательное выполнение данной функции мы и видим в консоли, после чего выводится итоговый результат).

После вывода заголовка Task8 показан результат вызова функции `Function8(int FunctionForCalculate(int), std::list<int> list)`. В итоге в консоль был выведен созданный ассоциативный массив, удовлетворяющий заданным в Таблице 1 требованиям.

ВЫВОД

В результате выполнения лабораторной работы №7 было разработано консольное приложение на языке программирования C++, реализующее заданные по варианту функции с применением принципов функционального программирования (в данном случае применено использование лямбда-функций).

Для удобства отображения необходимой информации в консоль были написаны вспомогательные функции `PrintMap(std::map<int, int> map)` и `PrintList(std::list<int> list)`.

ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММЫ

TRPO LR7.cpp

```
#include <iostream>
#include <list>
#include <map>

#pragma region Task2
//Функция для создания нового списка путем преобразования элементов списка, полученного в
качестве параметра
//Принимает 2 аргумента:
//    FunctionForCalculate(int) - функция для вычисления значения, зависящего от элемента списка
//    list - список для получения (путем применения FunctionForCalculate(int) к его элементам)
нового списка
//Возвращает:
//    новый список, полученный путем преобразования элементов списка, полученного в качестве
параметра, с помощью функции, полученной в качестве параметра
std::list<int> Function2(int FunctionForCalculate(int), std::list<int> list)
{
    std::list<int> newListOfNumbers;

    for (std::list<int>::iterator i = list.begin(); i != list.end(); i++)
    {
        newListOfNumbers.push_back(FunctionForCalculate(*i));
    }

    return newListOfNumbers;
}

//Функция для вывода списка в консоль
//Принимает 1 аргумент:
//    list - список для вывода
//Ничего не возвращает
void PrintList(std::list<int> list)
{
    for (std::list<int>::iterator i = list.begin(); i != list.end(); i++)
    {
        std::cout << *i << " ";
    }
}
#pragma endregion

#pragma region Task3
//Функция, вычисляющая результат, зависящий от предыдущего значения функции, полученной в
качестве параметра, и следующего элемента списка, полученного в качестве параметра
//Принимает 2 аргумента:
//    FunctionForCalculate(int, int) - функция для получения значения, зависящего от элементов
списка
//    list - список для получения (путем применения FunctionForCalculate(int, int) к его
элементам) значения, зависящего от его элементов
//Возвращает:
//    целое значение, зависящее от предыдущего значения функции, полученно в качестве параметра,
и следующего элемента списка, полученного в качестве параметра
int Function3(int FunctionForCalculate(int, int), std::list<int> list)
{
    int param1 = 0;

    if (list.size() < 3)
    {
        std::cout << "There are not enough items in the list";
        return 0;
    }
}
```



```

std::list<int>::iterator i = list.begin();
param1 = FunctionForCalculate(*i, *(i++));
std::cout << param1 << std::endl;
for (++i; i != list.end(); i++)
{
    param1 = FunctionForCalculate(param1, *i);
    std::cout << param1 << std::endl;
}

return param1;
}
#pragma endregion

#pragma region Task8
//Функция для генерации ассоциативного массива где:
//    ключ – значение функции, полученной в качестве параметра, от элемента списка, полученного в
качестве параметра
//    значение – количество таких значений.
//Принимает 2 аргумента:
//    FunctionForCalculate(int) - функция для вычисления значения, зависящего от элемента списка
//    list - список для получения (путем применения FunctionForCalculate(int) к его элементам)
ключа для ассоциативного массива
//Возвращает:
//    ассоциативный массив где:
//    ключ – значение функции, полученной в качестве параметра, от элемента списка,
полученного в качестве параметра
//    значение – количество таких значений.
std::map<int, int> Function8(int FunctionForCalculate(int), std::list<int> list)
{
    std::map<int, int> newMap;

    for (std::list<int>::iterator i = list.begin(); i != list.end(); i++)
    {
        newMap[FunctionForCalculate(*i)]++;
    }

    return newMap;
}

//Функция для вывода map в консоль
//Принимает 1 аргумент:
//    map - ассоциативный массив для вывода
//Ничего не возвращает
void PrintMap(std::map<int, int> map)
{
    for (std::map<int, int>::iterator i = map.begin(); i != map.end(); i++)
    {
        std::cout << "key: " << i->first << "\tvalue: " << i->second << std::endl;
    }
}
#pragma endregion

int main()
{
    std::list<int> originalListOfNumbers{ 2, 3, 2, 5, 4, 5 };
    std::list<int> currentListOfNumbers;
    std::map<int, int> map;

    std::cout << "Original list = { ";
    PrintList(originalListOfNumbers);

```

```

std::cout << "}\n\n";

std::cout << "=====Task 2=====\\n";
currentListOfNumbers = Function2(IncrementElement, originalListOfNumbers);
std::cout << "Original list after transform = { ";
PrintList(currentListOfNumbers);
std::cout << "}\n\n";

std::cout << "=====Task 3=====\\n";
std::cout << "result = " << Function3(SumOfTwoIntElements, originalListOfNumbers) <<
std::endl;

std::cout << "=====Task 8=====\\n";
map = Function8(IncrementElement, originalListOfNumbers);
std::cout << "generated map:\\n";
PrintMap(map);
}

```