МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

Лабораторная работа по теме

«Функциональное программирование в C++»

Цель: закрепить навыки функционального программирования на языке C++.

Теория: Функциональная парадигма – это декларативная парадигма программирования. Главное отличие декларативного программирования от императивного (например, структурного или объектно-ориентированного) заключается в процессе описания логику вычислений, но не указывая последовательности управляющих конструкций для описания вычислительного процесса. Одним из основных свойств языка функционального программирования является то, что все функции, описываемые в таком языке, являются объектами первого класса. Объект первого класса — это сущность, которая может быть динамически создана, уничтожена, передана в функцию, возвращена как значение и имеет все права, которыми обладают другие переменные. Функции в C++ не являются объектами первого класса, но ими являются экземпляры классов, указатели на функции, и (в некотором роде) анонимные функции. Таким образом можно утверждать, что C++ является мультипарадигменным языком программирования, в котором присутствуют элементы функционального программирования.

1) Сами по себе объекты в C++ имеют мало что общего с функциями, однако можно создать функциональный объект или функтор, перегрузив оператор круглых скобок. Таким образом можно объявить

экземпляр класса, принимающий аргументы, как и обычная функция, но при этом обладающую внутренним состоянием.

```
class sum {
   int x;
public:
   sum(int val) : x(val) {}
   int operator()(int y) const { return x + y; }
};
sum sum10(10);
int i = sum10(15);
```

2) Указатель на функции — это переменная, хранящая адрес функции, которая впоследствии может быть вызвана через этот указатель функции.

```
void func(int x){
    printf( "%d\n", x );
}
int main(){
    void (*pfunc)(int);
    pfunc = &func;
    return 0;
}
```

3) Анонимная функция или лямбда-функция — это функция без имени, которую можно встроить в исходном коде, обычно для передачи в другую функцию в качестве аргумента.

Например

```
[](int a, int b) { return a < b; }
```

где

`[]` – список захвата

`()` – список параметров

`{}` – код функции

Описание:

Выполните три задания по варианту применяя принципы функционального программирования

1) Напишите функцию, возвращающую новый список только из тех элементов списка-аргумента, для которых функция-аргумент возвращает значение true.
2) Напишите функцию, возвращающую новый список из значений функции-аргумента, примененной к элементам списка-аргумента.
3) Напишите функцию, возвращающую значение, полученное путем применения функции-аргумента от двух значений к предыдущему результату полученному этой функцией и следующему элементу списка-аргумента.
4) Напишите функцию `f(a)(b)(c)`… которая возвращает `a+b+c`…
5) Напишите функцию, вычисляющую время выполнения другой функции
6) Напишите функцию, которая применяет функцию к значению n раз
7) Напишите функцию проверяющую, являются ли все скобки в строке закрытыми
8) Напишите функцию, генерирующую ассоциативный массив, где ключ – значение функции-аргумента от элемента списка-аргумента, а значение – количество таких значений.
9) Напишите функцию для вычисления значения функции для всех элементов структуры данных «Дерево».

Варианты

№	*1*	*2*	*3*	*№*	*1*	*2*	*3*	*№*	*1*	*2*	*3*
1	2	4	9	*13*	9	2	3	*25*	9	2	7
2	3	8	1	*14*	8	3	2	*26*	5	4	8
3	8	7	3	*15*	3	8	9	*27*	2	3	8
4	9	3	1	*16*	2	9	8	*28*	8	7	9
5	4	8	9	*17*	8	2	4	*29*	1	9	4
6	6	2	7	*18*	1	5	6	*30*	9	4	3
7	8	5	9	*19*	2	4	5	*31*	5	2	3
8	8	6	2	*20*	2	8	4	*32*	7	5	1
9	7	5	2	*21*	6	4	7	*33*	8	3	4
10	9	7	5	*22*	4	2	8	*34*	6	3	1
11	4	5	9	*23*	2	1	4	*35*	9	7	4
12	4	3	1	*24*	6	9	4	*36*	2	7	9