|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
|  |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт кибербезопасности и цифровых технологий |
|  |

**Практическая работа № 4,**

по дисциплине: **«Технологии и методы программирования»**.

Выполнил студент 3 курса группы БИСО-01-21 Нагибин А. А.

Проверил преподаватель Лесько С.А.

Москва 2024

# Итератор

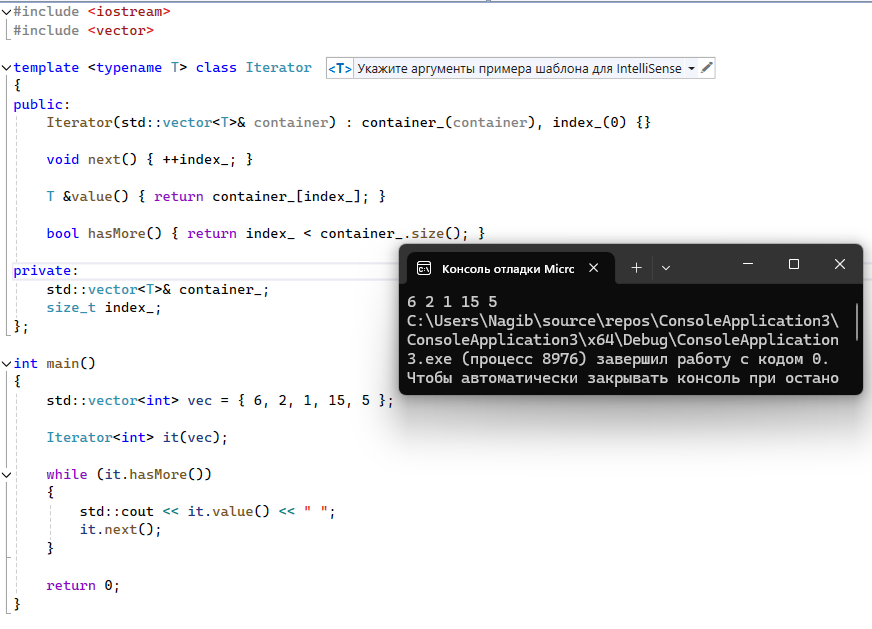


Рисунок 1 – листинг программы и результат работы

Листинг программы:

#include <iostream>

#include <vector>

template <typename T> class Iterator

{

public:

Iterator(std::vector<T>& container) : container\_(container), index\_(0) {}

void next() { ++index\_; }

T &value() { return container\_[index\_]; }

bool hasMore() { return index\_ < container\_.size(); }

private:

std::vector<T>& container\_;

size\_t index\_;

};

int main()

{

std::vector<int> vec = { 6, 2, 1, 15, 5 };

Iterator<int> it(vec);

while (it.hasMore())

{

std::cout << it.value() << " ";

it.next();

}

return 0;

}

# Посетитель

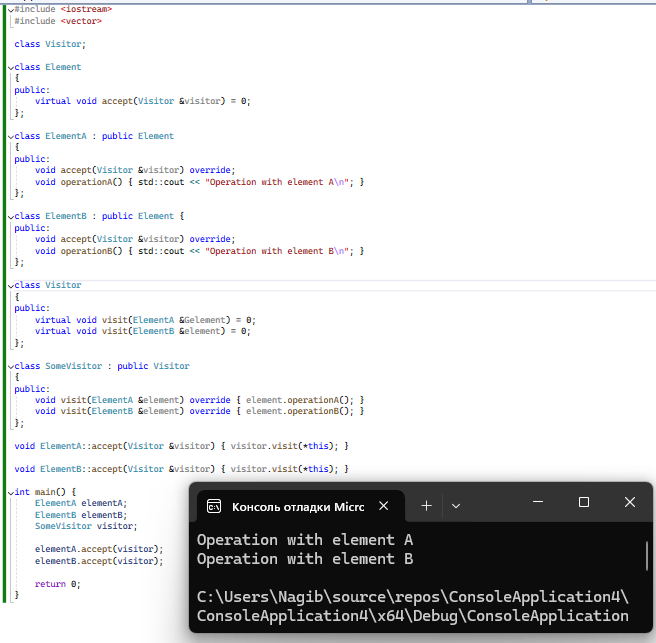


Рисунок 2 – листинг программы и результат работы

Листинг программы:

#include <iostream>

#include <vector>

class Visitor;

class Element

{

public:

virtual void accept(Visitor &visitor) = 0;

};

class ElementA : public Element

{

public:

void accept(Visitor &visitor) override;

void operationA() { std::cout << "Operation with element A\n"; }

};

class ElementB : public Element {

public:

void accept(Visitor &visitor) override;

void operationB() { std::cout << "Operation with element B\n"; }

};

class Visitor

{

public:

virtual void visit(ElementA &Gelement) = 0;

virtual void visit(ElementB &element) = 0;

};

class SomeVisitor : public Visitor

{

public:

void visit(ElementA &element) override { element.operationA(); }

void visit(ElementB &element) override { element.operationB(); }

};

void ElementA::accept(Visitor &visitor) { visitor.visit(\*this); }

void ElementB::accept(Visitor &visitor) { visitor.visit(\*this); }

int main() {

ElementA elementA;

ElementB elementB;

SomeVisitor visitor;

elementA.accept(visitor);

elementB.accept(visitor);

return 0;

}