Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №9 дисциплины
«Разработка программных систем»

Применение C-кода в языке программирования Python

Выполнил студент группы ИВТ-31_____/Крючков И. С./
Проверил_____/Чистяков Г. А./

Киров 2023

1. Цель

Целью работы является получение навыков повышения производительности приложений на Python за счет реализации вычислительно сложных операций в модулях на языке C/C++.

2. Задание

Для выполнения лабораторной работы необходимо решить следующие задачи:

- По результатам лабораторной работы №1 выбрать один из наиболее требовательных методов
- Реализовать данный метод на языке C/C++
- Оформить реализацию в виде подключаемого модуля
- Провести сравнения реализаций метода на Python и на C/C++ для набора подготовленных данных.

3. Листинг программы

Листинг программной реализации приведен в приложении А.

4. Результаты тестов

Для сравнения реализаций был выбран метод факторизации чисел.

Результаты приведены в таблице 1.

Таблица 1 – Результаты тестирования

№	Python, с	C++, с
1	0.18	0.0019
2	0.99	0.0098
3	1.13	0.0120
4	5.95	1.5700

5. Вывод

В ходе выполнения лабораторной работы были изучены повышения производительности программ на Python за счет реализации вычислительно сложных операций в модулях на языке C/C++.

Был разработан подключаемый модуль на языке C++ выполняющий факторизацию целых чисел. Выполнено сравнение производительности вычислений алгоритма, реализованного на python и алгоритма в подключаемом C++ модуле. По результатам время выполнения алгоритма, реализованного на C++ значительно ниже по сравнению с реализацией на Python.

Приложение А.

Листинг программы

main.py

```
from time import time
from fast_factorize import factorize
import sys

sys.setrecursionlimit(2000)

def main():
    data = [21990232406568, 219982384065688, 2199882324665688, 219988232466568884]

    def fact(n):
        i = 2
        prims = []
        while i * i <= n:
            while n % i == 0:
                prims.append(i)
                n /= i

            i += 1
        if n > 1:
            prims.append(n)
        return prims

    for i, v in enumerate(data):
        print(i+1)
        start = time()
        rp = fact(v)
        duration = time() - start
        print(f"[Python]: {time() - start}")

        r = factorize(v)
        print()


if __name__ == "__main__":
    main()
```

module.cpp

```
#include <Windows.h>
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
#include <iostream>
#include <chrono>
#include <string>

std::vector<long long> factorize(long long n) {
    long long i = 2;


    auto begin = std::chrono::high_resolution_clock::now();

    std::vector<long long> res = {1};

    while (i * i  <= n) {
        while (n % i == 0) {
            res.push_back(i);
            n /= i;
        }
        ++i;
    }

    if (n > 1)
        res.push_back(n);

    auto end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<float> elt = end - begin;

    std::cout << "[C++]: " << std::to_string(elt.count()) << " sec" << std::endl;

    return res;
}

namespace py = pybind11;

PYBIND11_MODULE(fast_factorize, m) {
    m.def("factorize", &factorize, "Factorize function");

#ifdef VERSION_INFO
    m.attr("__version__") = VERSION_INFO;
#else
    m.attr("__version__") = "dev";
#endif
}
```