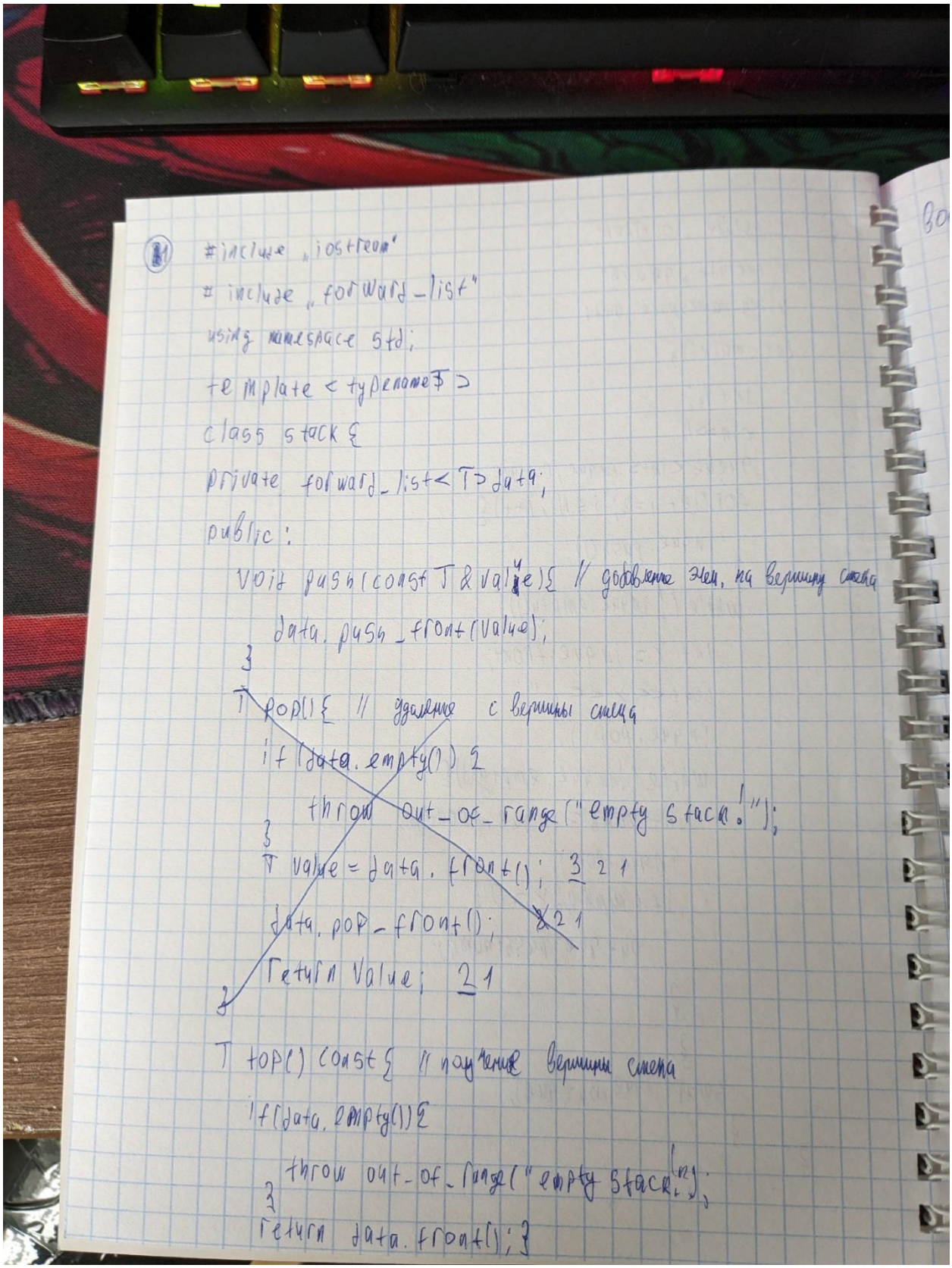
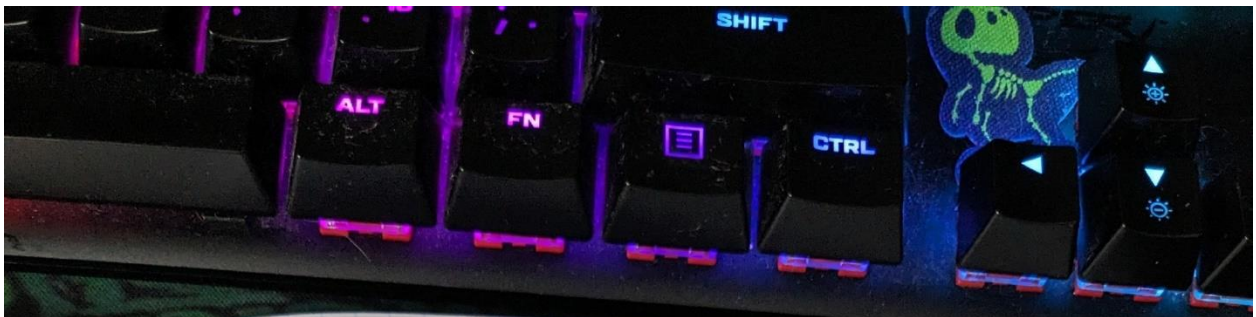
**Задание 1)** Реализуйте АТД Стек на односвязном списке (forward_list).



```
#include „iostream"
# include „forward_list"
using namespace std;
template <typename T>
class stack {
private forward_list<T> data;
public:
  void push(const T &value){ // добавление эле, на вершину стека
     data.push_front(value);
  }
  pop(){ // удаление с вершины стека
    if(data.empty()) {
       throw out_of_range("empty stack!");
    }
    T value = data.front(); 3 2 1
    data.pop_front(); 2 1
    return value; 2 1
  }
  T top() const{ // получение вершины стека
    if(data.empty()){
       throw out_of_range("empty stack!");
    }
    return data.front(); }
```




```
bool empty() const{
    return data.empty();
}
void pop(){
    if(data.empty()){
        throw out_of_range("empty stack!");
    }
    data.pop_front();
}

size_t size() const{ // forward_list не определяет ф-ии,
                     которые позволяют получить размер
                     контейнера, поэтому
                     метод проходится по
                     каждому элементу списка
                     и увеличивает счетчик.
                     Думаю это не очень эффективно,
                     можно было бы использовать
                     итераторы, но не уверен,
                     что это было бы лучше.)
    size_t count=0
    for(const auto& elem: data){
        count++;
    }
    return count;
}

};
```

**Задание 2**) Используя класс queue из STL решите следующую задачу.
(2736) Вывести простые числа среди чисел от 2 до $N$, используя следующий алгоритм:
Первоначально очередь все числа от 2 до $N$.

1. Взять первый элемент $X$ из входной очереди и напечатать.
2. В выходную очередь поместить числа из очереди, которые не кратны $X$.
3. Поменять входную и выходную очередь (swap).
4. Пока очередь не пуста, то повторять дествия с шага 2.

Ввод содержит одно целое число $N$ ($2 \le N \le 100000$).

② 

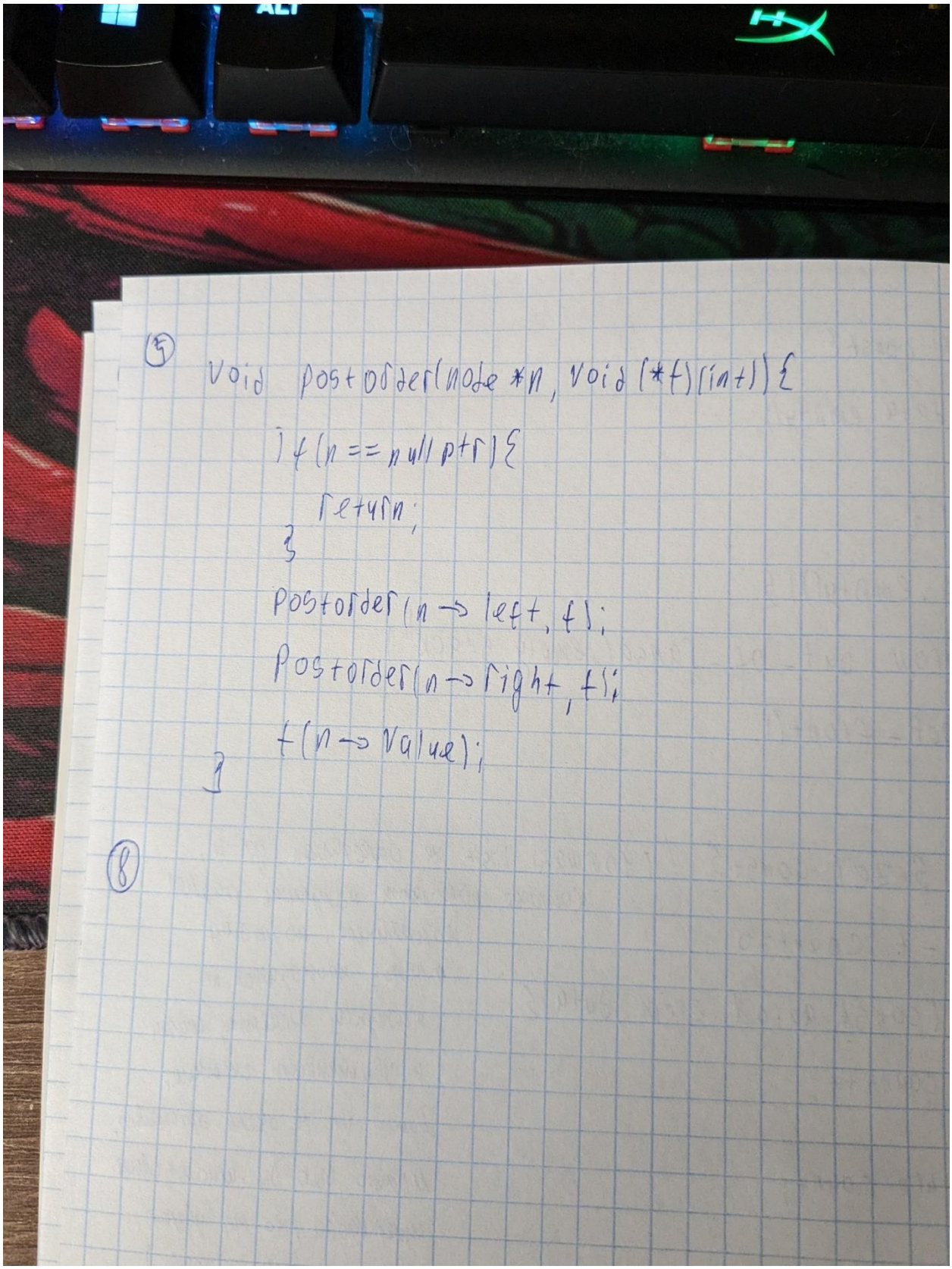
```
#include "iostream"
#include "queue"
using namespace std;
int main(){
    int N;
    cin>>N;
    queue<int> inque, outque;
    for(int i=2; i<=N; i++){
        inque.push(i);
    }
    while(!inque.empty()){
        int X = inque.front;
        cout<<X<<" ";
        inque.pop();
        while(!inque.empty()){
            int numb = inque.front();
            inque.pop();
            if(numb%X!=0){
                outque.push(numb);
            }
        }
        swap(inque, outque);
    }
}
```

**Задание 5)** Напишите функцию для обратного (post-order) обхода бинарного дерева, заданного следующей структурой:

```
struct node { int value; node *left, *right; };
```

К каждому значению, хранящемуся в дереве, функция применяет функцию, указанную в качестве аргумента:

```
void postorder(node *n, void (*f)(int))
```



**Задание 8)** Используя map из STL напишите решение следующей задачи с эффективностью O(NlogN).
Дана последовательность из n целых чисел. Найти непрерывную подпоследовательность максимальной длины, в которой нет одинаковых элементов. Вывести длину и начальный индекс найденной подпоследовательности.

1 prac.cpp *

```cpp
#include <iostream>
#include <vector>
#include <map>

using namespace std;

pair<int, int> findlong(const vector<int>& nums) {
    map<int, int> lastIndex;
    int maxLength = 0;
    int startIdx = 0;
    int left = 0;

    for (int right = 0; right < nums.size(); ++right) {
        if (lastIndex.find(nums[right]) != lastIndex.end() && lastIndex[nums[right]] >= left) {
            left = lastIndex[nums[right]] + 1; // сдвиг левого указателя вправо
        }
        lastIndex[nums[right]] = right; // обновляем последний индекс появления элемента
        if (right - left + 1 > maxLength) {
            maxLength = right - left + 1;
            startIdx = left;
        }
    }

    return {maxLength, startIdx};
}

int main() {
    vector<int> nums = {5, 1, 3, 5, 2, 3, 4, 1}; // exmp
    pair<int, int> result = findlong(nums);
    cout << "Длина: " << result.first << "\nНачальный индекс: " << result.second << endl;
    return 0;
}
```

**Задание 9)** Сравните время работы set и unordered_set из STL для операций поиска с количеством элементов N=100,10000,10^6,10^7 (например, измерить время поиска 10 существующих значений в наборе и 10 несуществующих). Ключами являются строки из случайных букв от a до z длиной ровно 16. Результат оформить в виде таблицы, время в ns. Привести код, использованный для измерения времени для одного значения N

```cpp
#include <iostream>
#include <set>
#include <unordered_set>
#include <string>
#include <chrono>
#include <random>

using namespace std;
using namespace std::chrono;

string get_random_string() {
    const string str = "abcdefghijklmnopqrstuvwxyz";
    string key;
    for (int i = 0; i < 16; i++)
        key += str[rand() % str.size()];
    return key;
}

int main() {
    const int N = 100;
    set<string> set;
    unordered_set<string> unordered_set;
    srand(time(NULL));
    for (int i = 0; i < N; i++) {
        string randomString = get_random_string();
        set.insert(randomString);
        unordered_set.insert(randomString);
    }
    auto start_t = high_resolution_clock::now();
    set.find("string");
    auto end_t = high_resolution_clock::now();
    auto time_set = duration_cast<nanoseconds>(end_t - start_t);
    start_t = high_resolution_clock::now();
    unordered_set.find("string");
    end_t = high_resolution_clock::now();
    auto time_u_set = duration_cast<nanoseconds>(end_t - start_t);
    cout << "Время работы set: " << time_set << endl;
    cout << "Время работы unordered_set: " << time_u_set << endl;

    return 0;
}
```

```
make: 'release/prac.exe' is up to date.
>Запуск программы...
Время работы set: 2700ns
Время работы unordered_set: 200ns
>Код завершения: 0 Время выполнения: 20 ms
Для продолжения нажмите любую клавишу . . .
```

```
>Запуск программы...
Время работы set: 3400ns
Время работы unordered_set: 300ns
>Код завершения: 0 Время выполнения: 20 ms
Для продолжения нажмите любую клавишу . . .
```

```
>Запуск программы...
Время работы set: 9100ns
Время работы unordered_set: 500ns
>Код завершения: 0 Время выполнения: 1910 ms
Для продолжения нажмите любую клавишу . . .
```

```
make: 'release/prac.exe' is up to date.
>Запуск программы...
Время работы set: 9800ns
Время работы unordered_set: 500ns
>Код завершения: 0 Время выполнения: 29040 ms
Для продолжения нажмите любую клавишу . . .
```

| A | B | C | D | E |
|---|---|---|---|---|
| N | Время работы set | Время работы unordered set | Спеки компа | |
| 100 | 2700ns | 200ns | Gpu: i3-12100f | 4 ядра 8 потоков 3.3ghz |
| 10000 | 3400ns | 300ns | | кэш l1 = 320 kb |
| 10^6 | 9100ns | 500ns | | кэш l2 = 5mb |
| 10^7 | 9800ns | 500ns | | кэш l3 = 12mb |
| | | | Ram: 3200mhz 8x2 GB | CAS latency = 16 |
| | | | | |
| | | | | |

**Задание 15)** Напишите функцию для проверки отсутствия циклов в орграфе, заданном через списки смежных вершин.

```
void dfs1(std::list<std::list<int>>& Graph, std::vector<int>& V, int index, int& count)
{
    V[index] = count++;
    for(auto i : Graph[index])
        if(V[i] == 0)
            dfs1(Graph, V, i, count);
}

void DFS(std::list<std::list<int>>& Graph, std::vector<int>& V)
{
    int count = 0;
    for(int i = 0; i < V.size(); i++)
        if(!V[i])
            dfs1(Graph, V, i, count);
}
```