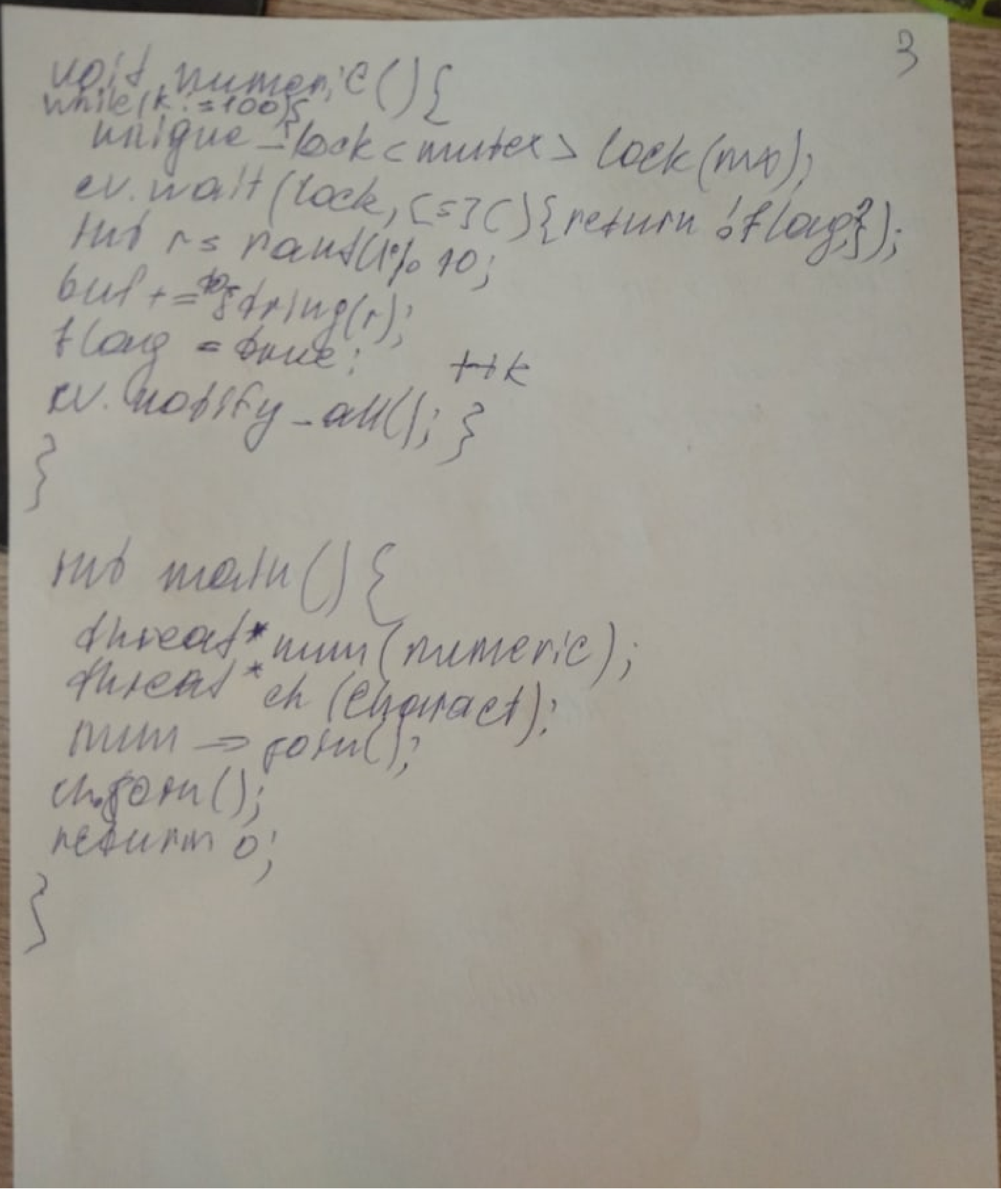

```
void numeric(){
while(k!=100){
  unique_lock<mutex> lock(mx);
  cv.wait(lock, [=](){return !flag;});
  int r= rand()%10;
  buf += to_string(r);
  flag = true;      ++k
  cv.notify_all(); }
}

int main(){
  thread* num(numeric);
  thread* ch (charact);
  num -> join();
  ch.join();
  return 0;
}
```

```
#include <map>
#include <string>
#include <algorithm>
#include <vector>
~~#include [illegible]~~

using namespase std;
class StringInfo{
   string str;
   map<char, int> data;
public:
   explicit StringInfo(string str){
       this -> str = str;
       countFreq(this->str);

   }
 StringInfo(const StringInfo &strInf){
     copy(strInfo.~~[illegible]~~str, begin, strInfo.str.end, back_inserter(str));
     this->data.insert(strInf.data.begin(), strInf.data
.end());
 }

void addSymbol(const string &symbols){
     this->str.append(symbols);
     countFreq(symbolols);
```

~~void print30()~~

```
map<string, int> year{...};
void print30(const map<string,int>
& y){

  for (map<string,int>::iterator it = [illegible]
                                          y.begin;
it != y.end; ++it){
  if (it->second ==30){
    cout << it->first << endl;
  }
}

void ~~print31~~ print31 (const map<string,int>
& y){

  for_each(y.begin, y.end, [](const pair
  <string, int>& pair){
     if (pair -> second == 31){
       cout << pair.->first << endl;
    });

   }
```

```
                                                  1
void countFreq(const string &substr){
    map<char,int> :: iterator itr;
    for(char c : substr){
    this->data[c]++;
   }
         <pair<char,int>
vector ~~char>~~ getMostPopular(){
   vector <pair<char,int>> pairs
   for(auto &itr : data){
        pairs.emplace_back(itr);
   }
   sort(pairs.begin, pairs.end(), [=](pair<char,
int>&a, pair<char,int>&b){
        return a.second < b.second;
   });
   vector<pair<char,int>> topFreq(pairs.
begin, pairs.begin+3);
    return topFreq;

} }

 };
```

}
void

```
# include<mutex>
# include <thread>
# include <iostream>
# include <string>
using namespace std;
mutex mx; int k=0;
condition_variable cv; bool flag = true;
string buf; charact
void ~~[illegible]~~()   {
while(k!=100){
   unique_lock<mutex> lock(mx);
   cv.wait(lock, [=](){return flag ~~[illegible]~~});
   char c;
   int r;
   srand(time(nullptr));
   r = rand() % 26;
   c = 'a'+r;
   ~~buf.append(c);~~
   buf += c; flag = false; ++k;
   ~~mx~~ cv.notify_all();
}
}
}
```