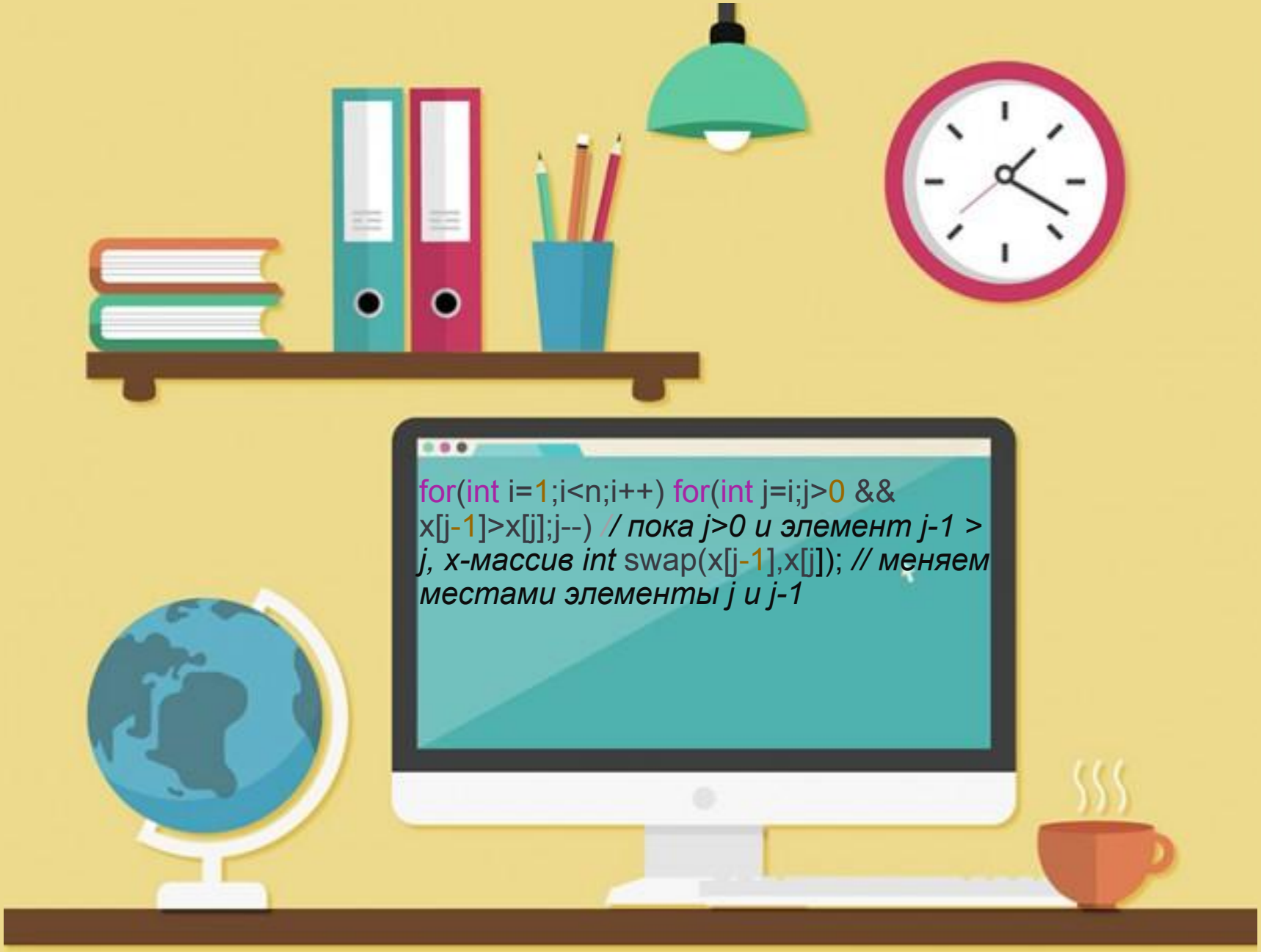
Сортировка вставка

Т
Т
Т
В
Д
К



```
for(int i=1;i<n;i++) for(int j=i;j>0 &&
x[j-1]>x[j];j--) // пока j>0 и элемент j-1 >
j, x-массив int swap(x[j-1],x[j]); // меняем
местами элементы j и j-1
```

Сортировка вставками имеет сложность n^2, количество сравнений вычисляется по формуле n*(n-1)

Для доказательства был написан следующий код:

```
void Sort(int* arr,int n){
        int counter=0;
        for(int i=1;i<n;i++){
                for(int j=i; j>0 && arr[j-1]>arr[j];j--){
                        counter++;
                        int tmp=arr[j-1];
                        arr[j-1]=arr[j];
                        arr[j]=tmp;
                }
        }
        cout<<counter<<endl;
}
```

Количество перестановок для 100 элементов:

Command Prompt

```
C:\Users\ART\Google Диск\sources\sort>a
Number of elements:100
Number of Swaps:4950

C:\Users\ART\Google Диск\sources\sort>
```

Таблица 1 — Сортировка вставками

шаг	отсортированная часть массива							тек. элемент	вставка
1	3							3	false
2	3	3						7	false
3	3	3	7					1	true
4	1	3	3	7				2	true
5	1	2	3	3	7			5	true
6	1	2	3	3	5	7		0	true
—	0	1	2	3	3	5	7	—	—

```cpp
// insertion_sort.cpp: определяет точку входа для консольного приложения.

#include "stdafx.h"
#include <iostream>
#include <ctime>
#include <iomanip>
using namespace std;

void insertionSort(int *, int); // прототип функции сортировки вставками

int main(int argc, char* argv[])
{
    srand(time(NULL));
    setlocale(LC_ALL, "rus");
    cout << "Введите размер массива: ";
    int size_array; // длинна массива
    cin >> size_array;

    int *sorted_array = new int [size_array]; // одномерный динамический массив
    for (int counter = 0; counter < size_array; counter++)
    {
        sorted_array[counter] = rand() % 100; // заполняем массив случайными числами
        cout << setw(2) << sorted_array[counter] << "  "; // вывод массива на экран
    }
    cout << "n";

    insertionSort(sorted_array, size_array); // вызов функции сортировки вставками

    for (int counter = 0; counter < size_array; counter++)
    {
        cout << setw(2) << sorted_array[counter] << "  "; // печать отсортированного массива
    }
    cout << "n";
    delete [] sorted_array; // высвобождаем память
    system("pause");
    return 0;
}

void insertionSort(int *arrayPtr, int length) // сортировка вставками
{
    int temp, // временная переменная для хранения значения элемента сортируемого массива
        item; // индекс предыдущего элемента
    for (int counter = 1; counter < length; counter++)
    {
        temp = arrayPtr[counter]; // инициализируем временную переменную текущим значением элемента ма
        item = counter-1; // запоминаем индекс предыдущего элемента массива
        while(item >= 0 && arrayPtr[item] > temp) // пока индекс не равен 0 и предыдущий элемент масси
        {
            arrayPtr[item + 1] = arrayPtr[item]; // перестановка элементов массива
            arrayPtr[item] = temp;
            item--;
        }
```