Министерство образования и науки Российской Федерации

1. Санкт-Петербургский Политехнический Университет Петра Великого
2. —
3. Институт компьютерных наук и кибербезопасности

**ЛАБОРАТОРНАЯ РАБОТА № 3**

1. «**Сортировки вставками массива и линейного списка**»
2. по дисциплине «Структуры данных»
   1. Выполнил
   2. студент гр. 5151001/40001 Кириллов Д.А.

<*подпись*>

* 1. Преподаватель
  2. асс. преподавателя Семьянов П.В.

<*подпись*>

1. Санкт-Петербург
2. 2025
3. **Цель.**

Проверить гипотезу, что сортировка вставками линейного быстрее сортировки вставками массива, используя функцию rtdsc

1. **Задачи.**
2. Реализовать сортировки вставками массива, линейного списка путем перестановки указателей и путем перестановки полей.
3. Произвести замеры количества тактов процессора, необходимых для этих сортировок, с помощью функции rtdsc. Сделать это с различным количеством входных чисел.
4. **Ход работы.**
   1. **Сортировки.**

Были реализованы 3 вида сортировок

1. Сортировка вставками массива:

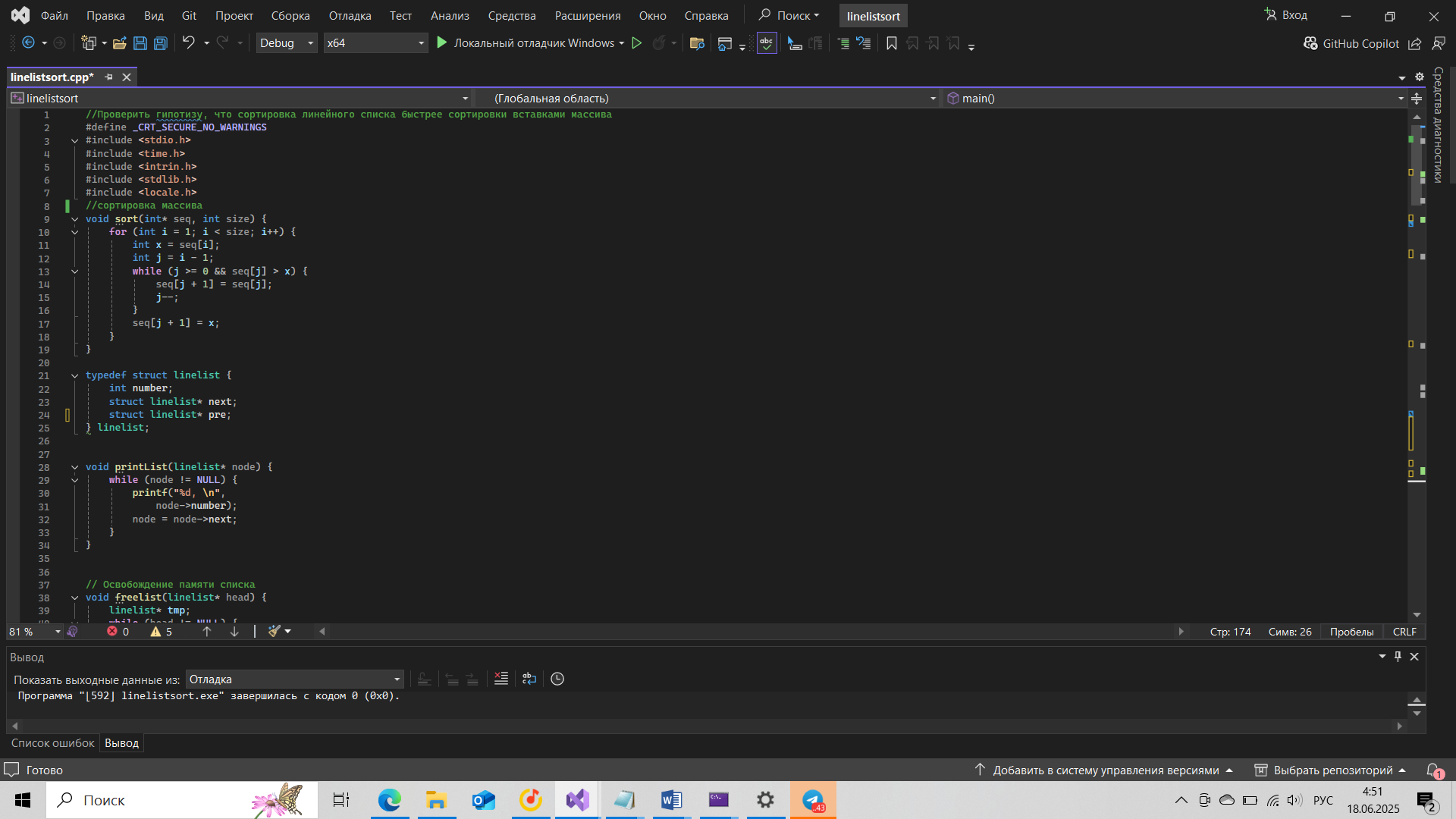


Рисунок 1 – сортировка массива вставками.

1. Сортировка вставками двусвязного линейного списка путем смены указателей:

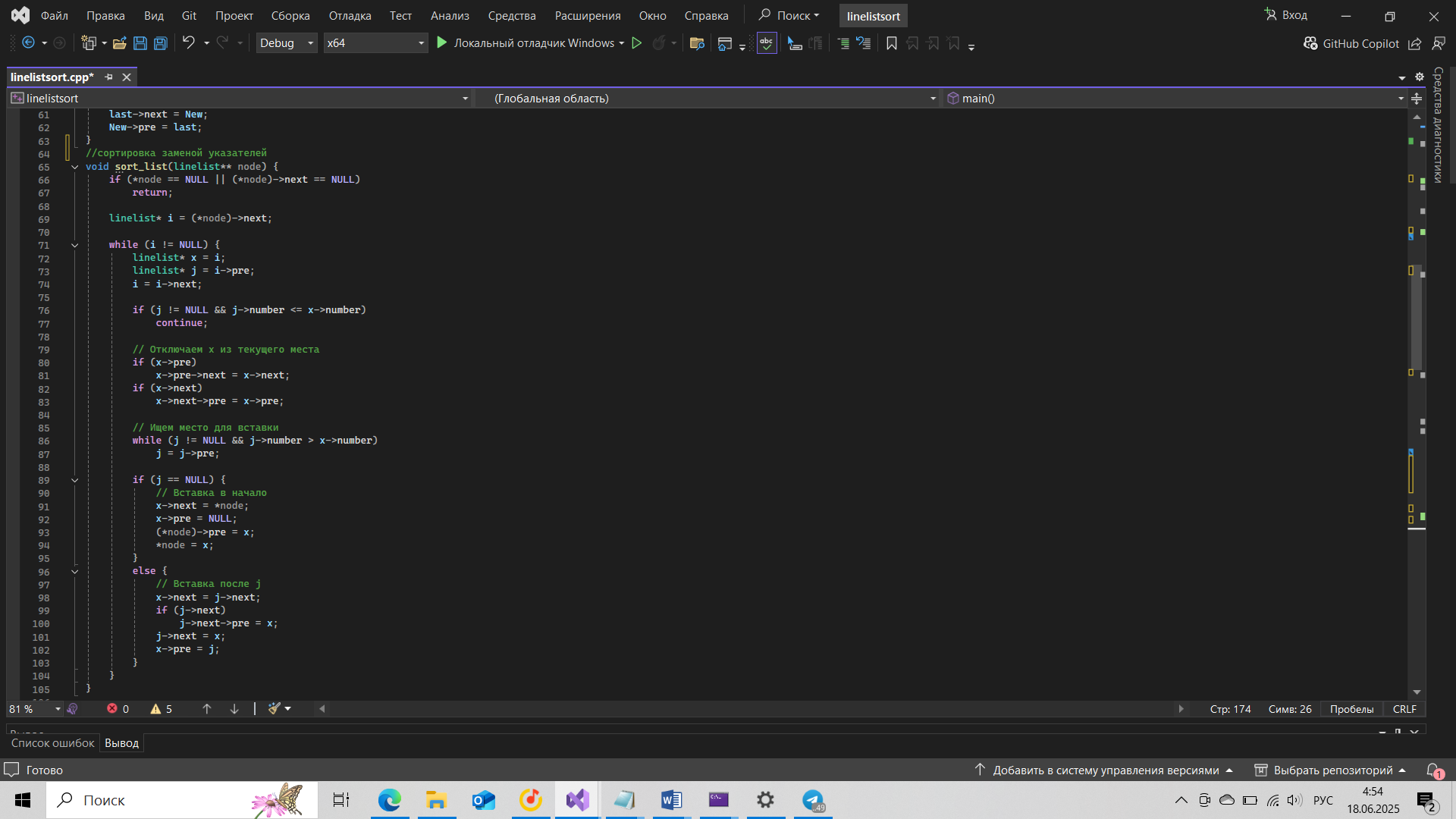


Рисунок 2 – сортировка линейного списка перестановкой указателей

1. Сортировка двусвязного списка путем замены значений в поле структуры линейного списка

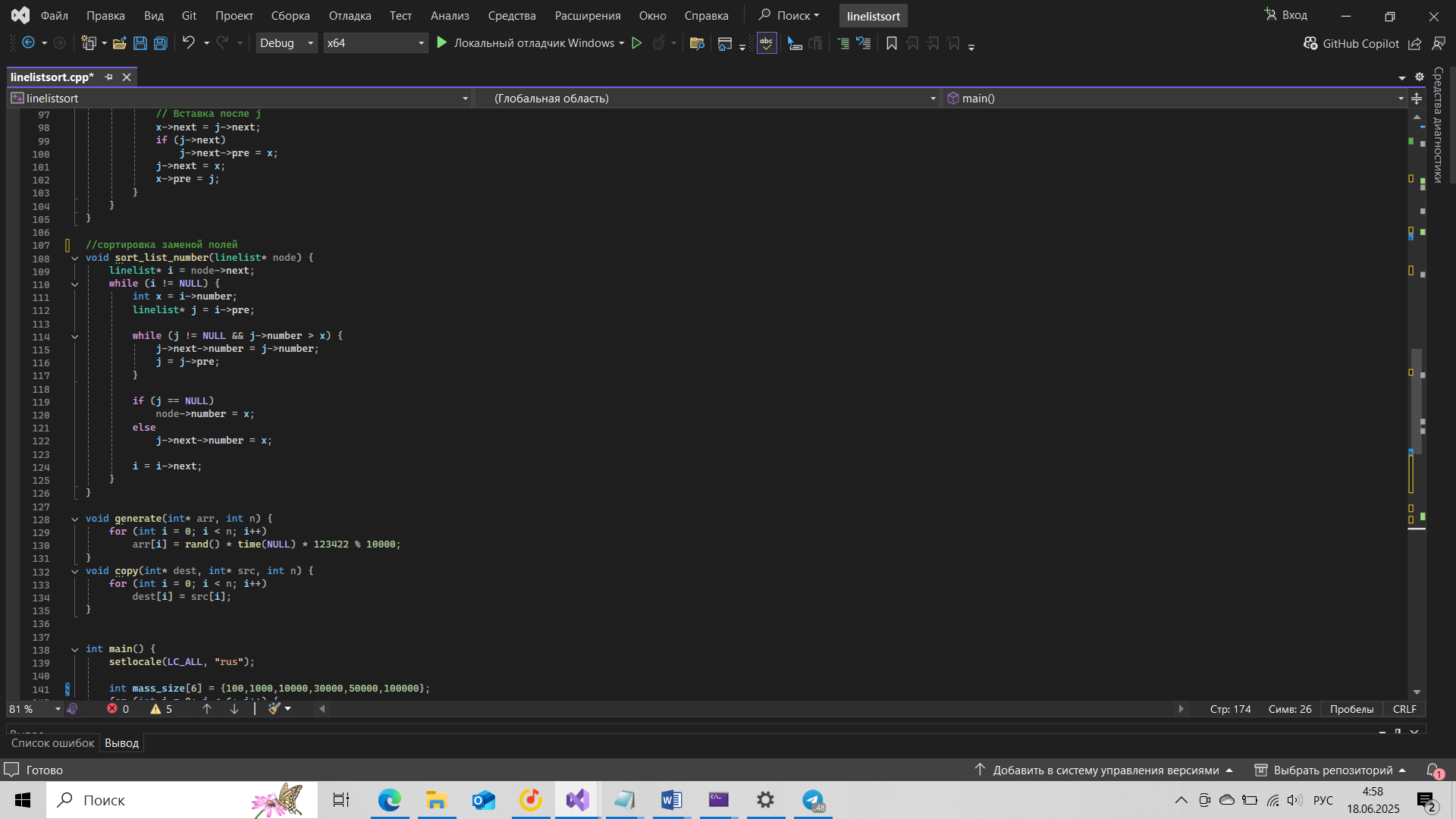


Рисунок 3 – сортировка линейного списка перестановкой полей

* 1. **Результаты.**

Сначала с помощью линейного конгруэнтного генератора сгенерировалось нужное количество значений (100,1000,10000,30000,50000,100000) и добавлялось в массив, после чего копировались в 2 линейных списка. После этого происходили замеры для массива и для списков. Замеры происходили на 10 последовательностях одинаковой длины для нахождения среднего количества тактов.

Таблица 1 – зависимость количества тактов, необходимых для сортировки от размера массива/списка.

|  |  |  |  |
| --- | --- | --- | --- |
| **Количество чисел в последовательности** | **Такты для массив** | **Такты для списка (значения)** | **Такты для списка (указатели)** |
| 100 | 5926 | 15367 | 13391 |
| 1000 | 507238 | 1397156 | 1195352 |
| 10000 | 33797359 | 150143847 | 311508013 |
| 30000 | 287329274 | 1353377269 | 4009152923 |
| 50000 | 811343809 | 3757726658 | 18204918440 |
| 100000 | 3397025317 | 15037514320 | 108143670805 |

Также было высчитано среднее время, необходимое для сортировки массива/списка по формуле , при частоте процессора 1900 МГц

Таблица 2 – зависимость времени, необходимого для сортировки от размера массива/списка.

|  |  |  |  |
| --- | --- | --- | --- |
| **Количество чисел в последовательности** | **Время для массива, с** | **Время для списка(поля), с** | **Время для списка(указатели),с** |
| 100 | 0 | 0 | 0 |
| 1000 | 0 | 0 | 0 |
| 10000 | 0,02 | 0,08 | 0,16 |
| 30000 | 0,15 | 0,71 | 2,11 |
| 50000 | 0,43 | 1,98 | 9,58 |
| 100000 | 1,79 | 7,91 | 56,92 |

1. **Вывод.**

Гипотеза о том, что сортировка линейного списка вставками работает быстрее сортировки вставками массива, не подтвердилась. Сортировка линейного списка путем смены указателей при небольших последовательностях (до 1000 включительно) занимает примерно одинаковое количество тактов, как и сортировка сменой полей, однако эти две сортировки все равно в несколько раз медленнее, чем сортировка массива. При больших последовательностях сортировка списка сменой указателей дольше до 6 раз, чем сменой полей и до 50 раз дольше сортировки массива.

**Приложение 1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <intrin.h>

#include <stdlib.h>

#include <locale.h>

//сортировка массива

void sort(int\* seq, int size) {

for (int i = 1; i < size; i++) {

int x = seq[i];

int j = i - 1;

while (j >= 0 && seq[j] > x) {

seq[j + 1] = seq[j];

j--;

}

seq[j + 1] = x;

}

}

typedef struct linelist {

int number;

struct linelist\* next;

struct linelist\* pre;

} linelist;

void printList(linelist\* node) {

while (node != NULL) {

printf("%d, \n",

node->number);

node = node->next;

}

}

// Освобождение памяти списка

void freelist(linelist\* head) {

linelist\* tmp;

while (head != NULL) {

tmp = head;

head = head->next;

free(tmp);

}

}

//вставка в массив

void insert(linelist\*\* node, int a) {

linelist\* New = (linelist\*)malloc(sizeof(linelist));

New->number = a;

New->next = NULL;

New->pre = NULL;

if (\*node == NULL) {

\*node = New;

return;

}

linelist\* last = \*node;

while (last->next != NULL) {

last = last->next;

}

last->next = New;

New->pre = last;

}

//сортировка заменой указателей

void sort\_list(linelist\*\* node) {

if (\*node == NULL || (\*node)->next == NULL)

return;

linelist\* i = (\*node)->next;

while (i != NULL) {

linelist\* x = i;

linelist\* j = i->pre;

i = i->next;

if (j != NULL && j->number <= x->number)

continue;

// Отключаем x из текущего места

if (x->pre)

x->pre->next = x->next;

if (x->next)

x->next->pre = x->pre;

// Ищем место для вставки

while (j != NULL && j->number > x->number)

j = j->pre;

if (j == NULL) {

// Вставка в начало

x->next = \*node;

x->pre = NULL;

(\*node)->pre = x;

\*node = x;

}

else {

// Вставка после j

x->next = j->next;

if (j->next)

j->next->pre = x;

j->next = x;

x->pre = j;

}

}

}

//сортировка заменой полей

void sort\_list\_number(linelist\* node) {

linelist\* i = node->next;

while (i != NULL) {

int x = i->number;

linelist\* j = i->pre;

while (j != NULL && j->number > x) {

j->next->number = j->number;

j = j->pre;

}

if (j == NULL)

node->number = x;

else

j->next->number = x;

i = i->next;

}

}

void generate(int\* arr, int n) {

for (int i = 0; i < n; i++)

arr[i] = rand() \* time(NULL) \* 123422 % 10000;

}

void copy(int\* dest, int\* src, int n) {

for (int i = 0; i < n; i++)

dest[i] = src[i];

}

int main() {

setlocale(LC\_ALL, "rus");

int mass\_size[6] = {100,1000,10000,30000,50000,100000};

for (int i = 0; i < 6; i++) {

int size = mass\_size[i];

int\* sequence = (int\*)malloc(size \* sizeof(int));

//int\* copy\_seq1 = (int\*)malloc(size \* sizeof(int));

//int\* copy\_seq2 = (int\*)malloc(size \* sizeof(int));

generate(sequence, size);

linelist\* node1 = NULL;

linelist\* node2 = NULL;

for (int j = 0; j < size; j++) {

insert(&node1, sequence[j]);

insert(&node2, sequence[j]);

}

//copy(copy\_seq1, sequence, size);

//copy(copy\_seq2, sequence, size);

// Тест сортировки массива

unsigned \_\_int64 start1 = \_\_rdtsc();

sort(sequence, size);

unsigned \_\_int64 end1 = \_\_rdtsc();

/\*linelist\* node1 = NULL;

linelist\* node2 = NULL;

for (int j = 0; j < size; j++) {

insert(&node1, copy\_seq1[j]);

insert(&node2, copy\_seq2[j]);

}\*/

// Тест сортировки списка меняя поля

unsigned \_\_int64 start2 = \_\_rdtsc();

sort\_list\_number(node1);

unsigned \_\_int64 end2 = \_\_rdtsc();

// Тест сортировки списка меняя указатели

unsigned \_\_int64 start3 = \_\_rdtsc();

sort\_list(&node2);

unsigned \_\_int64 end3 = \_\_rdtsc();

// Вывод результатов

printf("Количество чисел: %d", size);

printf("\nСортировка массива: %llu тактов\n", end1 - start1);

printf("Сортировка списка изменением полей: %llu тактов\n", end2 - start2);

printf("Сортировка списка сменой указателей: %llu тактов\n", end3 - start3);

printf("-------------------\n");

// Освобождение памяти

freelist(node1);

freelist(node2);

}

return 0;}