**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Лабораторная работа №7**

**по дисциплине «Языки программирования»**

Указатель на функцию

Группа: **АВТ-615**

Студент: **Кузенков В.С.**

Преподаватель: **Балагуров М.В.**

НОВОСИБИРСК 2017

# Постановка задачи

# Преобразовать функцию сортировки с использованием массивов (b), списков (6.3), деревьев (8.4, 8.5) в итератор. Проверить его работу на двух структурах, данных содержащих указатели на различные типы (например, целые и строки). Массив преобразовать в массив указателей.

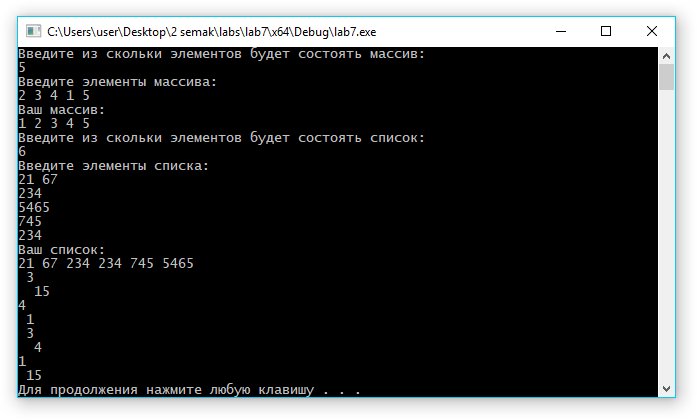
# Ограничения

Отсутствуют

# Тестирование

**Входные данные**

**Выходные данные**

****

# Текст программы:

#include <iostream>

#include <complex>

/\*Преобразовать функцию сортировки с использованием массивов (b), списков (6.3), деревьев (8.4, 8.5) в итератор. Проверить его работу на двух структурах данных содержащих указатели на различные типы (например, целые и строки). Массив преобразовать в массив указателей.\*/

using namespace std;

struct list

{

int val;

list \*prev = NULL;

list \*next = NULL;

};

struct node

{

int val;

node \*left = NULL;

node \*right = NULL;

node \*parent = NULL;

int num = 0;

};

void \*list\_first(void \*in)

{

return (list\*)in;

}

void \*list\_last(void \*in)

{

list \*temp = (list\*)in;

while(true)

{

if (((list\*)in)->next == NULL) break;

in = ((list\*)in)->next;

}

temp = (list\*)in;

while (((list\*)in)->prev != NULL)

{

in = ((list\*)in)->prev;

}

return temp;

}

void \*list\_next(void \*in)

{

return ((list\*)in)->next;

}

int list\_cmp(void \*first, void \*second)

{

return ((list\*)first)->val - ((list\*)second)->val > 0 ? 1 : -1;

}

void list\_swap(void \*first, void \*second)

{

int temp = ((list\*)first)->val;

((list\*)first)->val = ((list\*)second)->val;

((list\*)second)->val = temp;

}

void \*arr\_first(void \*in)

{

return (int\*)in + 1;

}

void \*arr\_last(void \*in)

{

return (int\*)(in)+\*(int\*)in;

}

void \*arr\_next(void \*in)

{

return (void\*)((int\*)(in) + 1);

}

int arr\_cmp(void \*first, void \*second)

{

return \*(int\*)first - \*(int\*)second > 0 ? 1 : -1;

}

void arr\_swap(void \*first, void \*second)

{

int temp = \*(int\*)first;

\*(int\*)first = \*(int\*)second;

\*(int\*)second = temp;

}

void \*arr\_perv(void \*in)

{

return (void\*)((int\*)(in)-1);

}

void sort(void\* struc, void \*begin(void\*), void \*end(void\*), void \*next(void\*), int cmp(void\*, void\*), void swap(void\*, void\*), void \*perve(void\*))

{

void \*i = begin(struc);

void \*last = end(struc);

i = next(i);

for (; ; i = next(i)) {

for (void \*j = i;cmp(j, perve(j)) < 0; j = perve(j))

{

swap(j, perve(j));

if (j == next(begin(struc))) break;

}

if (i == last) break;

}

}

void print\_list(list \*ls)

{

cout << "Ваш список:\n";

while(true)

{

cout << ls->val << ' ';

if (ls->next == NULL) break;

ls = ls->next;

}

cout << endl;

}

void print\_tree(node \*root, int level)

{

if (root)

{

if (root->left)

print\_tree(root->left, level + 1);

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

cout << " ";

cout << root->val << '\n';

if (root->right)

print\_tree(root->right, level + 1);

}

}

void print\_arr(int\* arr)

{

cout << "Ваш массив:\n";

for (int i = 1; i <= arr[0]; i++)

{

cout << arr[i] << " ";

}

cout << endl;

}

list\* input\_list(list \*ls)

{

cout << "Введите из скольки элементов будет состоять список: \n";

int size = 0;

cin >> size;

cout << "Введите элементы спискa: \n";

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

{

cin >> ls->val;

if(i == size-1)

{

break;

}

ls->next = new list;

ls->next->prev = ls;

ls = ls->next;

}

while(ls->prev != NULL)

{

ls = ls->prev;

}

return ls;

}

void input\_tree(node \*nd)

{

cout << "Введите из скольки элементов будет состоять дерево: \n";

int size = 0;

cin >> size;

}

int\* input\_arr(int \*arr)

{

cout << "Введите из скольки элементов будет состоять массив: \n";

int size = 0;

cin >> size;

arr = (int\*)realloc(arr,sizeof(int)\*(size+1));

arr[0] = size;

cout << "Введите элементы массива: \n";

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

{

cin >> arr[i];

}

return arr;

}

void \*list\_perv(void \*in)

{

return ((list\*)in)->prev;

}

void \*node\_first(void \*in)

{

node \*temp = (node\*)in;

while(true)

{

if (((node\*)temp)->parent == NULL) break;

temp = ((node\*)temp)->parent;

}

return (node\*)temp;

}

void \*node\_last(void \*in)

{

node \*temp = (node\*)in;

while (true)

{

if (((node\*)in)->right == NULL) break;

in = ((node\*)in)->right;

}

temp = (node\*)in;

while (((node\*)in)->parent != NULL)

{

in = ((node\*)in)->parent;

}

return temp;

}

void scan\_node(void\* in, int val, void \*&res)

{

if(in){

if (((node\*)in)->num == val)

{

res = in; // нашли!!!

}

scan\_node(((node\*)in)->left, val,res); // рекурсивный поиск влево

scan\_node(((node\*)in)->right, val,res);// рекурсивный поиск вправо

}

}

void \*node\_next(void \*in)

{

//cur\_num++;

node \*first = (node \*)node\_first(in);

void \*next = in;

scan\_node(first, ((node\*)in)->num + 1, next);

return (node\*)next;

}

void \*node\_perv(void \*in)

{

//cur\_num--;

void \*perv = in;

scan\_node((node\*)node\_first(in), ((node\*)in)->num - 1, perv);

return (node\*)perv;

}

int num = 1;

void node\_numming(node\* root) { //нумеруем в глубину

if (root) {

root->num = num++;

node\_numming(root->left);

node\_numming(root->right);

}

}

void print\_node\_numming(node\* root, int level) {

if (root)

{

if (root->left)

print\_node\_numming(root->left, level + 1);

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

cout << " ";

cout << root->num << '\n';

if (root->right)

print\_node\_numming(root->right, level + 1);

}

}

int node\_cmp(void \*first, void \*second)

{

return ((node\*)first)->val - ((node\*)second)->val > 0 ? 1 : -1;

}

void node\_swap(void \*first, void \*second)

{

int temp = ((node\*)first)->val;

((node\*)first)->val = ((node\*)second)->val;

((node\*)second)->val = temp;

}

int main()

{

setlocale(0, "rus");

//массив

int \*arr = new int;

arr = input\_arr(arr);

sort((void\*)arr, arr\_first, arr\_last, arr\_next, arr\_cmp, arr\_swap,arr\_perv);

print\_arr(arr);

delete arr;

//список

list \*ls = new list;

ls = input\_list(ls);

sort((void\*)ls, list\_first, list\_last, list\_next, list\_cmp, list\_swap,list\_perv);

print\_list(ls);

delete ls;

//дерево

node\* tree = new node;

tree->left = new node;

tree->right = new node;

tree->val = 4;

tree->left->val = 3;

tree->left->parent = tree;

tree->right->val = 1;

tree->right->parent = tree;

tree->left->right = new node;

tree->left->right->val = 15;

tree->left->right->parent = tree->left;

print\_tree(tree, 0);

node\_numming(tree);

sort((void\*)tree, node\_first, node\_last, node\_next, node\_cmp, node\_swap, node\_perv);

print\_tree(tree, 0);

delete tree->left;

delete tree->right;

delete tree;

system("pause");

return 0;

}

**Вывод:** Указатели на функцию – мощный и полезный инструмент, позволяющий, к примеру, реализовать полиморфизм при помощи итераторов для разных типов структур.