Звіт з лабораторної роботи
за дисципліною "алгоритми і структури даних"
студента групи ПА-17-1
Панасенка Єгора Сергійовича
Кафедра комп’ютерних технологій, фпм, дну
2017-2018 навч.р.

1. Постановка задачі:

   Розробити програму, в якій

   a) Реалізувати сортування шаблонами

   b) Зробити сортування списків двома способами: заміною вузлів або їх значень

   c) Порівняти швидкість сортування масиву та списків.

2. Опис структури програми та реалізованих функцій:

   a) Програма задає struct element по шаблону від класу T з такими полями:

      1. T value – значення елемента
      2. element<T> * prev – посилання на попередній елемент.
      3. element<T> * next – посилання на наступний елемент.

   b) Програма задає клас List по шаблону від класу T такими полями:

      1. element<T> * head – перший елемент списку
      2. element<T> * last – останній елемент списку
      3. int size – розмір списку

      та такими приватними функціями:

      4. void DeleteNode (element<T> * node)
         1. Видаляє елемент зі списку
         2. Аргументи:
            1. node – посилання на елемент у списку
         3. Функція нічого не виводить.
      5. element<T> * FindByKey (int key)
         1. Знаходить елемент по номеру
         2. Аргументи:

    1. key – номер елемента у списку
  3. Функція виводить адресу елемента.
6. element<T> * FindByValue (T value)
  1. Знаходить елемент по значенню
  2. Аргументи:
    1. value – номер елемента у списку
  3. Функція виводить адресу елемента.
7. void SwapNodes (element<T> * node1, element<T> * node2)
  1. Міняє місцями два елементи
  2. Має таку послідовність дій:
    1. Зберігаються усі адреси елементів
    2. Адреси міняються місцями
    3. Попередити попередні та наступні елементи про зміну
    4. Якщо вийшло що якийсь з двох елементів дивиться сам на себе, то позначити адресу на другий елемент.
  3. Аргументи:
    1. node1 – адреса першого елемента
    2. node2 – адреса другого елемента
  4. Функція нічого не виводить.

та такими публічними функціями:

8. List () - конструктор, створює пустий список.
9. ~List() - деструктор, видаляє усі елементи.
10. void Clear()
  1. Видаляє усі елементи
  2. Функція не має аргументів та нічого не виводить.
11. void Add (T value, int position = 0)
  1. Додає елемент до списку з можливістю додавати з кінця, за допомогою від’ємної позиції (додавання в кінець константна складність), якщо не вказано позицію, додає в початок
  2. Аргументи:
    1. value – значення яке потрібно додати
    2. position – позиція у яку потрібно додати

3. Функція нічого не виводить

12. int Size() - функція виводить розмір списку
13. int Get(int key, T * value)
    1. Забирає значення елемента зі списку
    2. Аргументи:
        1. key – номер елемента
        2. value – адреса у яку записують значення елемента
    3. Функція виводить успіх знаходження
14. void DeleteByKey (int key)
    1. Видаляє елемент по номеру
    2. Аргументи:
        1. key – номер елемента
    3. Функція нічого не виводить
15. void DeleteByValue (T value)
    1. Видаляє елемент по значенню
    2. Аргументи:
        1. value – значення елемента
    3. Функція нічого не виводить
16. void print(ostream& out)
    1. Виводить список у поток
    2. Аргументи:
        1. out – вихідний поток
    3. Функція нічого не виводить
17. void Swap (int key1, int key2)
    1. Міняє місцями два елемента по номеру
    2.
    3. Аргументи:
        1. key1 – номер першого елемента
        2. key2 – номер другого елемента
    4. Функція нічого не виводить
18. void Sort ()
    1. Сортує елементи за степеневу складність (n^2) замінюючи вузли

2. Функція не має аргументів та нічого не виводить.

19. void SortByKey ()
    1. Сортує елементи за степеневу складність (n^2) замінюючи значення
    2. Функція не має аргументів та нічого не виводить.
20. T * getArray ()
    1. Конвертує список у масив та виводить його
21. void setArray (T * array, int array_size)
    1. Конвертує массив у список
    2. Аргументи:
        1. array – вхідний масив
        2. array_size -розмір масиву
    3. Функція нічого не виводить

c) Программа має такі функції:

1. template <typename T> ostream& operator<<(ostream& out, List<T>& obj)
    1. Перевантаження змінної виходу, виводить список через пробіл
    2. Аргументи:
        1. out – вихідний поток
        2. obj – список, який потрібно вивести
    3. Виводить вихідний поток
2. template <typename T> istream& operator>>(istream& in, List<T>& obj)
    1. Перевантаження змінної входу, забирає список
    2. Аргументи:
        1. in – вхідний поток
        2. obj – список, у який потрібно зберегти
3. template <class T> void sort_array (T * array, int size)
    1. Сортує масив елементів
    2. Аргументи:
        1. array – вхідний масив
        2. size -розмір масиву

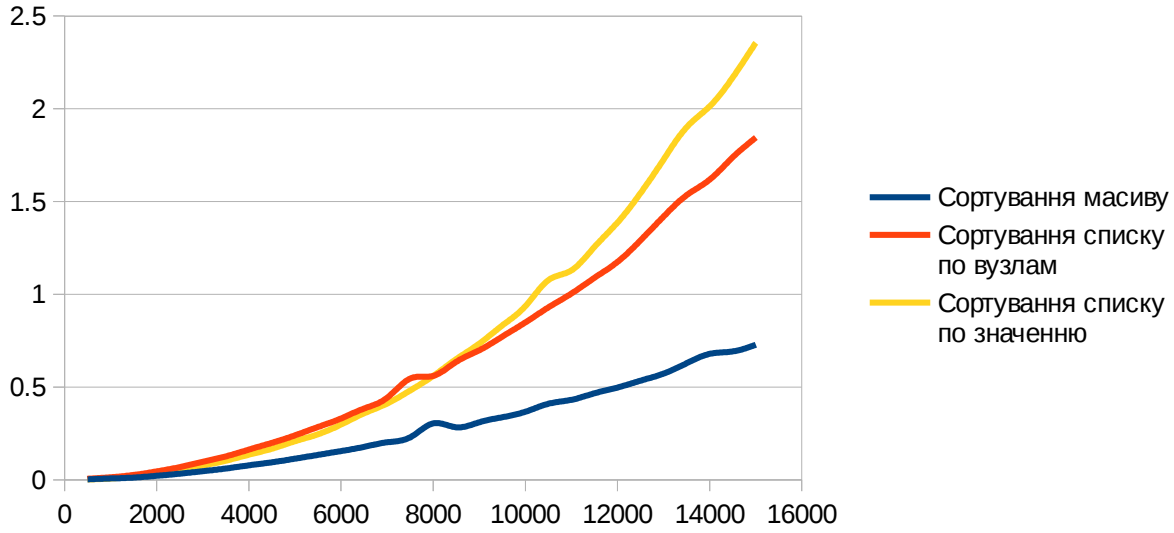
3. Функція нічого не виводить

4. int get_number (istream& in, ostream& out, string message, int min = INT_MIN, int max = INT_MAX)
   1. Забирає число у потрібному діапазоні
   2. Аргументи:
      1. in – вхідний поток
      2. out – вихідний поток
      3. message – повідомлення про число яке потрібно ввести
      4. min – мінімально можливе число
      5. max – максимально можливе число
   3. Виводить число
5. template <int> int random (int min, int max)

   template <class T> double random (int min, int max)
   1. Виводить випадкове число
   2. Аргументи:
      1. min – мінімально можливе число
      2. max – максимально можливе число
6. template <class T> void menu (istream& in, ostream& out, int * state)
   1. Виводить меню на экран, тут описано головні дії програми
   2. Аргументи:
      1. in – вхідний поток
      2. out – вихідний поток
      3. state – адреса на статус у якій працює програма (int, double)
   3. Функція нічого не виводить
7. int main()
   1. Головна функція, закускає меню у потрібному режимі.

3. Порівняння швидкостей сортування (у секундах), таблиця та графік:

Розмір	Сортування масиву	Сортування списку по вузлам	Сортування списку по значенню
500	0.00258	0.005479	0.001643
1000	0.006967	0.013381	0.00676
1500	0.011689	0.025679	0.017436
2000	0.021248	0.045051	0.033285
2500	0.032684	0.068169	0.05273
3000	0.046228	0.09694	0.076492
3500	0.060553	0.125848	0.102779
4000	0.077843	0.162745	0.135002
4500	0.094073	0.19886	0.167371
5000	0.113567	0.239289	0.207664
5500	0.134067	0.284453	0.245312
6000	0.154658	0.330287	0.297752
6500	0.177235	0.384138	0.358007
7000	0.201813	0.441815	0.41079
7500	0.228504	0.545588	0.481085
8000	0.303887	0.55996	0.559453
8500	0.282929	0.635552	0.650536
9000	0.309826	0.697991	0.733699
9500	0.336913	0.77178	0.832096
10000	0.366462	0.848007	0.935767
10500	0.409716	0.929272	1.075911
11000	0.430945	1.003841	1.128637
11500	0.466184	1.089193	1.25516
12000	0.497129	1.174487	1.386837
12500	0.534262	1.290929	1.545298
13000	0.572428	1.419146	1.726544
13500	0.62804	1.534076	1.900516
14000	0.6785	1.618178	2.013022
14500	0.69183	1.738416	2.168563
15000	0.727714	1.844506	2.35623



4. Код:

main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <climits>
#include <cstdlib>
#include <ctime>

using namespace std;

template <class T>
struct element
{
	T value;
	struct element * prev, * next;
};

template <class T>
class List {
	element<T> * head, * last;
	int size;
private:
	void DeleteNode (element<T> * node) {
		if (node == NULL) return;
		if (node->prev == NULL)
			head = node->next;
		else node->prev->next = node->next;
		if (node->next == NULL)
			last = node->prev;
		else node->next->prev = node->prev;
		size--;
		delete node;
	}
	element<T> * FindByKey (int key) {
		if (head == NULL || key < 0 || key >= size)
			return NULL;
		element<T> * node = head;
		for (int i = 0; i < key; i++)
			node = node->next;
		return node;
	}
	element<T> * FindByValue (T value) {
		if (head == NULL)
			return NULL;
		element<T> * node = head;
		while (node != NULL) {
			if (node->value == value) return node;
			node = node->next;
		}
		return NULL;
	}
	void SwapNodes (element<T> * node1, element<T> * node2) {
		if (node1 == NULL || node2 == NULL) return;

		element<T> * node1prev, * node1next, * node2prev, * node2next;
```

```
            node1prev = node1->prev;
            node1next = node1->next;
            node2prev = node2->prev;
            node2next = node2->next;

            if (node1prev != NULL)
                  node1prev->next = node2;
            else head = node2;

            if (node1next != NULL)
                  node1next->prev = node2;
            else last = node2;

            if (node2prev != NULL)
                  node2prev->next = node1;
            else head = node1;

            if (node2next != NULL)
                  node2next->prev = node1;
            else last = node1;

            node1->prev = node2prev;
            node1->next = node2next;
            node2->prev = node1prev;
            node2->next = node1next;

            if (node1->next == node1)
                  node1->next = node2;
            if (node1->prev == node1)
                  node1->prev = node2;
            if (node2->next == node2)
                  node2->next = node1;
            if (node2->prev == node2)
                  node2->prev = node1;
      }
      void SwapNodesByKey (element<T> * node1, element<T> * node2) {
            if (node1 == NULL || node2 == NULL) return;
            T tmp = node1->value;
            node1->value = node2->value;
            node2->value = tmp;
      }
public:
      List () {
            head = NULL;
            last = NULL;
            size = 0;
      }
      ~List() {
            Clear();
      }
      void Clear() {
            if (head == NULL)
                  return;
            element<T> * iter = head, * next;
            while (iter != NULL) {
                  next=iter->next;
                  delete iter;
```

```
            iter=next;
      }
      head = NULL;
      last = NULL;
      size = 0;
}
void Add (T value, int position = 0) {
      element<T> * node =  new element<T>;
      node->value = value;
      node->next = NULL;
      node->prev = NULL;
      int pos = position % (size + 1);
      if (size == 0) {
            head = node;
            last = node;
            size++;
            return;
      }
      element<T> * prev = NULL, * next = NULL;
      if (pos >= 0) {
            next = head;
            for (int i = 0; i < pos; i++) {
                  prev = next;
                  next = next->next;
            }
      } else {
            prev = last;
            for (int i = pos; i < -1; i++) {
                  next = prev;
                  prev = prev->prev;
            }
      }
      if (prev != NULL) {
            prev->next = node;
            node->prev = prev;
      }
      else head = node;
      if (next != NULL) {
            next->prev = node;
            node->next = next;
      }
      else last = node;
      size++;
}
int Size() {
      return size;
}
int Get(int key, T * value) {
      element<T> * node = FindByKey(key);
      if (node == NULL)
            return 0;
      *value = node->value;
      return 1;
}
void DeleteByKey (int key) {
      DeleteNode(FindByKey(key));
}
void DeleteByValue (T value) {
```

```
	DeleteNode(FindByValue(value));
}
void print(ostream& out) {
	if (head == NULL) {
		out << "Список пустий\n";
		return;
	} else if (size > 300) {
		out << "Список занадто великий\n";
		return;
	}
	element<T> * iter = head;
	while (iter->next != NULL) {
		out << iter->value << " ";
		iter=iter->next;
	}
	out << iter->value << endl;
}

void Swap (int key1, int key2) {
	element<T> * node1, * node2;
	node1 = FindByKey(key1);
	node2 = FindByKey(key2);
	SwapNodes(node1,node2);
}

void Sort () {
	if (head == last) return;
	for (element<T> * i = head; i != NULL; i=i->next) {
		for (element<T> * j = i->next; j != NULL; j=j->next) {
			if (i->value>j->value) {
				SwapNodes(i,j);
				element<T> * tmp = i;
				i = j;
				j = tmp;
			}
		}
	}
}

void SortByKey () {
	if (head == last) return;
	for (element<T> * i = head; i != NULL; i=i->next)
		for (element<T> * j = i->next; j != NULL; j=j->next)
			if (i->value>j->value)
				SwapNodesByKey(i,j);
}

T * getArray () {
	T * array = (T*) calloc(size,sizeof(T));
	element<T> * iter = head;
	for (int i = 0; i < size; i++) {
		array[i] = iter->value;
		iter = iter->next;
	}
	return array;
}

void setArray (T * array, int array_size) {
```

```
            Clear();
            for (int i = 0; i < array_size; i++) {
                  Add(array[i],-1);
            }
      }
};

template <typename T> ostream& operator<<(ostream& out, List<T>& obj)
{
      obj.print(out);
      return out;
}
template <typename T> istream& operator>>(istream& in, List<T>& obj)
{
      std::string::size_type x;
      string line;
      getline(in,line);
      while ((x = line.find(' ')) != std::string::npos) {
            x = line.find(' ');
            if (x != 0)
                  obj.Add(atoi(line.substr(0,x).c_str()),-1);
            line.erase(0,x+1);
      }
      obj.Add(atoi(line.c_str()),-1);
      return in;
}

template <class T>
void sort_array (T * array, int size) {
      T tmp;
      for (int i = 0; i < size; i++)
            for (int j = i + 1; j < size; j++)
                  if (array[i] > array[j]) {
                        tmp = array[i];
                        array[i] = array[j];
                        array[j] = tmp;
                  }
}

int get_number (istream& in, ostream& out, string message, int min = INT_MIN, int
max = INT_MAX) {
      int number = min-1, flag = 0;
      if (min == INT_MIN) flag = 1;
      string line;
      while (number < min || number > max || flag) {
            out << message << ":\n";
            getline(in,line);
            number = atoi(line.c_str());
            if (number < min || number > max) {
                  system("clear");
                  out << "Вы ввели неправильне число, дозволено тільки ціле число
від " << min << " до " << max << "\n";
            }
            if (flag == 1) flag = 0;
      }
      return number;
}
```

```
template <int>
int random (int min, int max) {
	return rand() % (max - min + 1) + min;
}

template <class T>
double random (int min, int max) {
	return (double(rand() % (max*10 - min*10 + 1) + min*10))/10;
}

template <class T>
void menu (istream& in, ostream& out, int * state) {
	List<T> list, list1;
	string line;
	int answer = 0, min, max, size;
	T x = 0, y = 0, * array;
	while (answer != 14 && answer != 15) {
		out << "Список має таку кількість елементів: " << list.Size() << endl;
		out << "Поточний список такий:\n";
		out << list;
		out << "Виберіть дію:\n";
		out << " 1. Додати рядок в кінець списку з клавіатури\n";
		out << " 2. Додати випадкові n елементів в кінець списку\n";
		out << " 3. Додати 50 випадкових елементів від -100 до 100\n";
		out << " 4. Додати елемент в потрібну позицію\n";
		out << " 5. Видалити один елемент зі списку за номером\n";
		out << " 6. Видалити один елемент зі списку за значенням\n";
		out << " 7. Вивести елемент за номером\n";
		out << " 8. Поміняти елементи місцями\n";
		out << " 9. Очищення списку\n";
		out << "10. Відсортувати список заміною вузлів\n";
		out << "11. Відсортувати список заміною значень\n";
		out << "12. Відсортувати список через масив\n";
		out << "13. Порівняти сорування массиву, списку (заміною вузлів та
заміною значень)\n";
		out << "14. Перейти у режим " << ((*state == 1)?"double":"int") <<
endl;
		out << "15. Вихід\n>>> ";
		getline(in,line);
		answer = atoi(line.c_str());
		system("clear");
		switch (answer) {
			case 1:
				out << "Список має таку кількість елементів: " <<
list.Size() << endl;
				out << "Введіть будь-які числа підряд\n";
				in >> list;
				break;
			case 2:
				x = get_number(in,out,"Введіть кількість",1);
				min = 1; max = 0;
				while (min > max) {
					min = get_number(in,out,"Введіть мінімально можливе
число");
					max = get_number(in,out,"Введіть максимально можливе
число");
					if (min > max) {
						out << "Неможливий діапазон\n";
```

```
                        }
                    }
                    for (int i = 0; i < x; i++) {
                        list.Add(random<T>(min,max),-1);
                    }
                    break;
                case 3:
                    x = 50;
                    min = -100; max = 100;
                    for (int i = 0; i < x; i++) {
                        list.Add(random<T>(min,max),-1);
                    }
                    break;
                case 4:
                    out << list;
                    x = get_number(in,out,"Введіть число");
                    y = get_number(in,out,"Введіть позицію (нуль - рандомна
позиція, відємне число - відрахвати з останього, позитивнне число - відрахувати з
першого)");
                    if (y > 0) y--;
                    else if (y == 0) y = rand();
                    list.Add(x, y);
                    break;
                case 5:
                    out << list;
                    list.DeleteByKey(get_number(in,out,"Введіть
число",1,list.Size()) - 1);
                    break;
                case 6:
                    out << list;
                    list.DeleteByValue(get_number(in,out,"Введіть число"));
                    break;
                case 7:
                    if (list.Get(get_number(in,out,"Введіть число",1) - 1,&x))
                        out << x << endl;
                    else out << "Елемент не існує" << endl;
                    break;
                case 8:
                    if (list.Size() > 0) {
                        out << list;
                        x = get_number(in,out,"Введіть номер",1,list.Size())
- 1;
                        y = get_number(in,out,"Введіть номер",1,list.Size())
- 1;
                        list.Swap(x,y);
                    }
                    break;
                case 9:
                    list.Clear();
                    break;
                case 10:
                    list.Sort();
                    break;
                case 11:
                    list.SortByKey();
                    break;
                case 12:
                    array = list.getArray();
```

```
                        size = list.Size();
                        sort_array(array,size);
                        list.setArray(array,size);
                        free(array);
                        break;
                  case 13:
                        T * array_orig, * array;
                        out << "N,Array,List1,List2\n";
                        clock_t t1,t2;
                        double dur[3];
                        for (int n = 500; n <= 20000; n += 500) {
                              array_orig = (T *) calloc(n,sizeof(T));
                              array = (T *) calloc(n,sizeof(T));
                              for (int i = 0; i < n; i++) {
                                    array_orig[i] = random<T>(-1000,1000);
                                    array[i] = array_orig[i];
                              }
                              t1 = clock();
                              sort_array(array,n);
                              t2 = clock();
                              dur[1] = 1.0*(t2-t1)/CLOCKS_PER_SEC;
                              list1.setArray(array_orig,n);
                              t1 = clock();
                              list1.Sort();
                              t2 = clock();
                              dur[2] = 1.0*(t2-t1)/CLOCKS_PER_SEC;
                              list1.setArray(array_orig,n);
                              t1 = clock();
                              list1.SortByKey();
                              t2 = clock();
                              dur[3] = 1.0*(t2-t1)/CLOCKS_PER_SEC;
                              out << n << "," << fixed << dur[1] << ","
                                    << fixed << dur[2] << ","
                                    << fixed  << dur[3] << endl;
                              list1.Clear();
                              free(array_orig);
                              free(array);
                        }
                        break;
                  case 14:
                        *state = (*state == 1) ? 2 : 1;
                        break;
                  case 15:
                        *state = 0;
                        break;
                  default:
                        out << "Неправильне значення, треба ввести число від 1 до
11\n";
            }
      }
      list.Clear();
      list1.Clear();
}

int main() {
      srand(time(NULL));
      int state = 1;
      while (state != 0) {
```

```
            if (state == 1)
                  menu<int>(cin, cout, &state);
            else menu<double>(cin, cout, &state);
      }
      return 0;
}
```

5. Результати роботи програми:

Приклад 1:

```
Список має таку кількість елементів: 0
Поточний список такий:
Список пустий
Виберіть дію:
 1. Додати рядок в кінець списку з клавіатури
 2. Додати випадкові n елементів в кінець списку
 3. Додати 50 випадкових елементів від -100 до 100
 4. Додати елемент в потрібну позицію
 5. Видалити один елемент зі списку за номером
 6. Видалити один елемент зі списку за значенням
 7. Вивести елемент за номером
 8. Поміняти елементи місцями
 9. Очищення списку
10. Відсортувати список заміною вузлів
11. Відсортувати список заміною значень
12. Відсортувати список через масив
13. Порівняти сорування массиву, списку (заміною вузлів та заміною значень)
14. Перейти у режим double
15. Вихід
>>> 3
Список має таку кількість елементів: 50
Поточний список такий:
28 79 -41 -20 10 64 -30 16 72 87 65 74 -21 77 -61 -90 -25 83 36 -63 14 76 20 -75
-25 55 3 22 -47 -31 91 81 -52 -93 -39 13 26 -14 -70 -1 -26 95 -71 7 28 -33 -26 58
-50 -90
Виберіть дію:
 1. Додати рядок в кінець списку з клавіатури
 2. Додати випадкові n елементів в кінець списку
 3. Додати 50 випадкових елементів від -100 до 100
 4. Додати елемент в потрібну позицію
 5. Видалити один елемент зі списку за номером
 6. Видалити один елемент зі списку за значенням
 7. Вивести елемент за номером
 8. Поміняти елементи місцями
 9. Очищення списку
10. Відсортувати список заміною вузлів
11. Відсортувати список заміною значень
12. Відсортувати список через масив
13. Порівняти сорування массиву, списку (заміною вузлів та заміною значень)
14. Перейти у режим double
15. Вихід
>>> 10
Список має таку кількість елементів: 50
Поточний список такий:
-93 -90 -90 -75 -71 -70 -63 -61 -52 -50 -47 -41 -39 -33 -31 -30 -26 -26 -25 -25
-21 -20 -14 -1 3 7 10 13 14 16 20 22 26 28 28 36 55 58 64 65 72 74 76 77 79 81 83
87 91 95
Виберіть дію:
```

```
 1. Додати рядок в кінець списку з клавіатури
 2. Додати випадкові n елементів в кінець списку
 3. Додати 50 випадкових елементів від -100 до 100
 4. Додати елемент в потрібну позицію
 5. Видалити один елемент зі списку за номером
 6. Видалити один елемент зі списку за значенням
 7. Вивести елемент за номером
 8. Поміняти елементи місцями
 9. Очищення списку
10. Відсортувати список заміною вузлів
11. Відсортувати список заміною значень
12. Відсортувати список через масив
13. Порівняти сорування массиву, списку (заміною вузлів та заміною значень)
14. Перейти у режим double
15. Вихід
>>> 14
Список має таку кількість елементів: 0
Поточний список такий:
Список пустий
Виберіть дію:
 1. Додати рядок в кінець списку з клавіатури
 2. Додати випадкові n елементів в кінець списку
 3. Додати 50 випадкових елементів від -100 до 100
 4. Додати елемент в потрібну позицію
 5. Видалити один елемент зі списку за номером
 6. Видалити один елемент зі списку за значенням
 7. Вивести елемент за номером
 8. Поміняти елементи місцями
 9. Очищення списку
10. Відсортувати список заміною вузлів
11. Відсортувати список заміною значень
12. Відсортувати список через масив
13. Порівняти сорування массиву, списку (заміною вузлів та заміною значень)
14. Перейти у режим int
15. Вихід
>>> 3
Список має таку кількість елементів: 50
Поточний список такий:
95.7 64.6 42.2 15.9 45.1 -83.4 -72.6 -51.2 -4.9 -20.1 -27 -57.7 -83.5 -23.5 -51.4
-67.6 90.4 74.7 18 76.1 -70.7 91.2 27.2 -86.4 -45.4 -44.1 -19.6 28 -85.6 30.4 37.8
65.8 -49.4 -20.1 -62.7 51.4 -47.8 -35.3 -99.9 47.4 0.4 -71.2 89.7 16.9 5.4 94 49.4
-48.6 24.4 -77
Виберіть дію:
 1. Додати рядок в кінець списку з клавіатури
 2. Додати випадкові n елементів в кінець списку
 3. Додати 50 випадкових елементів від -100 до 100
 4. Додати елемент в потрібну позицію
 5. Видалити один елемент зі списку за номером
 6. Видалити один елемент зі списку за значенням
 7. Вивести елемент за номером
 8. Поміняти елементи місцями
 9. Очищення списку
10. Відсортувати список заміною вузлів
11. Відсортувати список заміною значень
12. Відсортувати список через масив
13. Порівняти сорування массиву, списку (заміною вузлів та заміною значень)
14. Перейти у режим int
15. Вихід
```

```
>>> 12
Список має таку кількість елементів: 50
Поточний список такий:
-99.9 -86.4 -85.6 -83.5 -83.4 -77 -72.6 -71.2 -70.7 -67.6 -62.7 -57.7 -51.4 -51.2
-49.4 -48.6 -47.8 -45.4 -44.1 -35.3 -27 -23.5 -20.1 -20.1 -19.6 -4.9 0.4 5.4 15.9
16.9 18 24.4 27.2 28 30.4 37.8 42.2 45.1 47.4 49.4 51.4 64.6 65.8 74.7 76.1 89.7
90.4 91.2 94 95.7
Виберіть дію:
 1. Додати рядок в кінець списку з клавіатури
 2. Додати випадкові n елементів в кінець списку
 3. Додати 50 випадкових елементів від -100 до 100
 4. Додати елемент в потрібну позицію
 5. Видалити один елемент зі списку за номером
 6. Видалити один елемент зі списку за значенням
 7. Вивести елемент за номером
 8. Поміняти елементи місцями
 9. Очищення списку
10. Відсортувати список заміною вузлів
11. Відсортувати список заміною значень
12. Відсортувати список через масив
13. Порівняти сорування массиву, списку (заміною вузлів та заміною значень)
14. Перейти у режим int
15. Вихід
>>> 15
```