МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания 2

Тема:

Технология реализации алгоритмов с использованием функций (процедурное программирование) в заданиях дисциплины. Статические и динамические массивы. Разработка операций.

Выполнил студент Цемкало А. Р.

Фамилия И.О.

группа ИКБО-10-20

Москва 2021

СОДЕРЖАНИЕ

Задание 1

Постановка задачи

Разработать программу выполнения операций над статическим массивом целых чисел. Размер массива 1000 элементов. Предусмотреть ввод значения n – текущий размер массива. Разработать операции для управления массивом и реализовать их функциями. Функции должны принимать входные данные через параметры и возвращать результат, если этого требует алгоритм операции.

1.1. Разработка функции ввода массива из n элементов с клавиатуры.

```
void fill_in(int *list, int n) {
	for (int i = 0; i < n; i++) {
		cin >> list[i];
	}
}
```

1.2. Разработать функцию вывода массива из n на монитор.

```
void print_massiv(int* list, int n) {
	for (int i = 0; i < n; i++) {
		cout << list[i];
		if (i < n - 1) {
			cout << " ";
		}
	}
}
```

1.3. Разработать функцию заполнения массива из n элементов, используя датчик случайных чисел.

```
void random_fill_in(int* list, int n) {
	for (int i = 0; i < n; i++) {
		list[i] = rand();
	}
}
```

1.4. Разработать функцию и протестировать работу функций.

```
int main() {
	int massiv[1000];
	int n;
	cin >> n;
	if (n < 1 || n > 1000) {
		cout << "Недопустимый размер массива";
		return 1;
	}
	fill_in(massiv, n);
	print_massiv(massiv, n);
	cout << endl;
	random_fill_in(massiv, n);
	print_massiv(massiv, n);
	return 0;
}
```

1.5. Разработать функцию поиска первого вхождения значения в массив.

```
int find_first_comer(int* list, int n, int x) {
```

```
	for (int i = 0; i < n; i++) {
		if (list[i] == x) {
			return i;
		}
	}
}

int main() {
	setlocale(0, "");
	int massiv[1000];
	int n;
	cin >> n;
	if (n < 1 || n > 1000) {
		cout << "Недопустимый размер массива";
		return 1;
	}
	fill_in(massiv, n);

	cout << endl;
	int index = find_first_comer(massiv, n, 3);
	if (index == -1) {
		cout << "Такого значения в массиве нет";
		return 1;
	}
	else {
		cout << index;
	}

	print_massiv(massiv, n);

	return 0;
}
```

Тестирование функции:

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Искомое значение: 5	4	4
2	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Искомое значение: 12	Такого значения в массиве нет	Такого значения в массиве нет
3	Кол-во элементов: 8 Массив: 1 2 3 1 2 3 4 5 Искомое значение: 3	2	2

1.6. Разработать функцию нахождения индекса первого отрицательного числа в массиве.

```
int find_first_negative(int* list, int n) {
	for (int i = 0; i < n; i++) {
		if (list[i] < 0) {
			return i;
		}
	}
```

```
	return -1;
}


int main() {
	setlocale(0, "");
	int massiv[1000];
	int n;
	cin >> n;
	if (n < 1 || n > 1000) {
		cout << "Недопустимый размер массива";
		return 1;
	}
	fill_in(massiv, n);

	cout << endl;
	int negative_number_index = find_first_negative(massiv, n);
	if (negative_number_index == -1) {
		cout << "Отрицательного числа в массиве нет";
		return 1;
	}
	else {
		cout << negative_number_index;
	}

	print_massiv(massiv, n);

	return 0;
}
```

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 8 Массив: 1 2 3 4 5 -6 7 8	5	5
2	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10	Отрицательного числа в массиве нет	Отрицательного числа в массиве нет
3	Кол-во элементов: 7 Массив: 1 2 0 -5 -1 8 -9	3	3

1.7. Разработать функцию поиска всех вхождений в массив.

```
int find_all_comers(int* list, int n, int x) {
	bool f = false;
	for (int i = 0; i < n; i++) {
		if (list[i] == x) {
			if (f != false) {
				cout << " ";

			}
			f = true;
			cout << i;
		}
	}
	if (f == false) {
		return -1;
	}
	else {
```

```
			return 0;
		}
}

int main() {
		setlocale(0, "");
		int massiv[1000];
		int n;
		cin >> n;
		if (n < 1 || n > 1000) {
			cout << "Недопустимый размер массива";
			return 1;
		}
		fill_in(massiv, n);

		cout << endl;
		int indexes = find_all_comers(massiv, n, 5);
		if (indexes == -1) {
			cout << "Такого значения в массиве нет";
			return 1;
		}

		print_massiv(massiv, n);

		return 0;
}
```

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 8 Массив: 1 2 3 1 2 3 4 5 Искомое значение: 3	2 5	2 5
2	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Искомое значение: 12	Такого значения в массиве нет	Такого значения в массиве нет
3	Кол-во элементов: 8 Массив: 1 2 3 1 2 3 4 5 Искомое значение: 5	7	7

1.8. Разработать функцию вставки нового значения в заданную позицию массива.

```
void insert_x(int* list, int& n, int i, int x) {
		n++;
		for (int j = n - 2; j >= i; j--) {
			list[j + 1] = list[j];
		}
		list[i] = x;
}
int main() {
		setlocale(0, "");
		int massiv[1000];
		int n;
		cin >> n;
		if (n < 1 || n > 1000) {
			cout << "Недопустимый размер массива";
```

```
		return 1;
	}
	fill_in(massiv, n);

	int number;
	int x;
	cin >> number >> x;
	if (number >= 0 and number < n) {
		insert_x(massiv, *&n, number, x);
	}
	else {
		cout << "Недопустимый индекс";
		return 1;
	}
	cout << endl; print_massiv(massiv, n);

	return 0;
}
```

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 8 Массив: 1 2 3 1 2 3 4 5 Индекс: 4 Новое значение: 7	1 2 3 1 7 2 3 4 5	1 2 3 1 7 2 3 4 5
2	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Индекс: 0 Новое значение: 7	7 1 2 3 4 5 6 7 8 9 10	7 1 2 3 4 5 6 7 8 9 10
3	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Индекс: 12 Новое значение: 7	Недопустимый индекс	Недопустимый индекс

1.9. Разработать функцию алгоритма удаления со сжатием из массива значения в заданной позиции, сохраняя порядок следования остальных элементов.

```
void delete_index(int* list, int &n, int index) {
	for (int i = index + 1; i < n; i++) {
		list[i - 1] = list[i];
	}
	n--;
}
int main() {
	setlocale(0, "");
	int massiv[1000];
	int n;
	cin >> n;
	if (n < 1 || n > 1000) {
		cout << "Недопустимый размер массива";
		return 1;
	}
	fill_in(massiv, n);
```

```
	int deleted_index;
	cin >> deleted_index;
	if (deleted_index >= 0 && deleted_index < n) {
		delete_index(massiv, *&n, deleted_index);
	}
	else {
		cout << "Недопустимый индекс";
		return 1;
	}


	cout << endl;
	print_massiv(massiv, n);

	return 0;
}
```

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 8 Массив: 5 8 9 4 6 2 1 7 Индекс: 2	5 8 4 6 2 1 7	5 8 4 6 2 1 7
2	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Индекс: 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
3	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Индекс: 10	Недопустимый индекс	Недопустимый индекс

1.10. Разработать функцию удаления со сжатием всех вхождений заданного значения из массива. Сложность алгоритма $O(n^2)$.

```
void delete_all_comers_n2(int* list, int& n, int x) {
	int i = 0;
	while (i < n) {
		if (list[i] == x) {
			for (int j = i; j < n - 1; j++) {
				list[j] = list[j + 1];
			}
			n--;
		}
		else {
			i++;
		}
	}
}
```

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 8	5 8 9 4 6 1 7	5 8 9 4 6 1 7

	Массив: 5 8 9 4 6 2 1 7 Удаляемое значение: 2		
2	Кол-во элементов: 7 Массив: 5 8 7 5 5 6 5 Удаляемое значение: 5	8 7 6	8 7 6
3	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Удаляемое значение: 11	1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10

Квадратичная зависимость времени от n появляется из-за того, что в алгоритме удаление происходит за счёт двух циклов, один из которых вложенный. Цикл while ищет элемент, который нужно удалить, а вложенный в него цикл for удаляет с сжатием этот элемент.

1.11. Разработать функцию удаления со сжатием всех вхождений заданного значения из массива. Сложность алгоритма O(n).

```
void delete_all_comers_n(int* list, int& n, int x) {
	int j = 0;
	for (int i = 0; i < n; i++) {
		list[j] = list[i];
		if (list[i] != x) {
			j++;
		}
	}
	n = j;
}
```

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 8 Массив: 5 8 9 4 6 2 1 7 Удаляемое значение: 2	5 8 9 4 6 1 7	5 8 9 4 6 1 7
2	Кол-во элементов: 7 Массив: 5 8 7 5 5 6 5 Удаляемое значение: 5	8 7 6	8 7 6
3	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Удаляемое значение: 11	1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10

Зависимость времени от n является линейной, потому что в функции есть только один цикл. Все нужные элементы записываются в начало массива, а удаляемые игнорируются, то есть происходит перезапись массива (поверх старого массива записывается новый с только нужными элементами).

Задание 2

Постановка задачи

Реализовать операции вставки и удаления значений в динамическом массиве, модифицируя соответствующие функции статического массива. Использовать функцию realloc модуля malloc при изменении размера массива. Выполнить тестирование операций изменения размера массива. Применить к массиву функции поиска, разработанные для статического массива.

2.1. Реализация операции вставки значения в динамическом массиве, модифицируя соответствующую функцию статического массива.

```
void insert_x(int** list, int i, int x) {
	*list = (int*)realloc(*list, (_msize(*list) / sizeof(int) + 1) * sizeof(int));
	for (int j = _msize(*list) / sizeof(int) - 2; j >= i; j--) {
		(*list)[j + 1] = (*list)[j];
	}

	(*list)[i] = x;
}
int main() {
	setlocale(0, "");

	int n;
	cin >> n;
	if (n < 1 || n > 1000) {
		cout << "Недопустимый размер массива";
		return 1;
	}

	int* massiv;
	massiv = (int*)malloc(n * sizeof(int));

	int x;
	int index;
	cin >> index >> x;
	if (index >= 0 && index < _msize(massiv) / sizeof(int)) {
		insert_x(&massiv, index, x);
	}
	else {
		cout << "Недопустимый индекс";
		return 1;
	}

	cout << endl;
	print_massiv(massiv);

	return 0;
}
```

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 8 Массив: 1 2 3 1 2 3 4 5 Индекс: 4 Новое значение: 7	1 2 3 1 7 2 3 4 5	1 2 3 1 7 2 3 4 5

2	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Индекс: 0 Новое значение: 7	7 1 2 3 4 5 6 7 8 9 10	7 1 2 3 4 5 6 7 8 9 10
3	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Индекс: 12 Новое значение: 7	Недопустимый индекс	Недопустимый индекс

2.2. Реализация операции удаления со сжатием из массива значения в заданной позиции, сохраняя порядок следования остальных элементов (для динамического массива, модифицируя соответствующую функцию статического массива).

```
void delete_index(int** list, int index) {
	for (int i = index + 1; i < _msize(*list) / sizeof(int); i++) {
		(*list)[i - 1] = (*list)[i];
	}
	*list = (int*)realloc(*list, (_msize(*list) / sizeof(int) - 1) * sizeof(int));
}
int main() {
	setlocale(0, "");

	int n;
	cin >> n;
	if (n < 1 || n > 1000) {
		cout << "Недопустимый размер массива";
		return 1;
	}

	int* massiv;
	massiv = (int*)malloc(n * sizeof(int));

	fill_in(massiv);

	int deleted_index;
	cin >> deleted_index;
	if (deleted_index >= 0 && deleted_index < _msize(massiv) / sizeof(int)) {
		delete_index(&massiv, deleted_index);
	}
	else {
		cout << "Недопустимый индекс";
		return 1;
	}

	cout << endl;
	print_massiv(massiv);

	return 0;
}
```

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы

1	Кол-во элементов: 8 Массив: 5 8 9 4 6 2 1 7 Индекс: 2	5 8 4 6 2 1 7	5 8 4 6 2 1 7
2	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Индекс: 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
3	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Индекс: 10	Недопустимый индекс	Недопустимый индекс

2.3. Реализация операции удаления со сжатием всех вхождений заданного значения из динамического массива, модифицируя соответствующую функцию статического массива. Сложность алгоритма $O(n^2)$.

```
void delete_all_comers_n2(int** list, int x) {
	int i = 0;
	while (i < _msize(*list) / sizeof(int)) {
		if ((*list)[i] == x) {
			for (int j = i; j < _msize(*list) / sizeof(int) - 1; j++) {
				(*list)[j] = (*list)[j + 1];
			}
			*list = (int*)realloc(*list, (_msize(*list) / sizeof(int) - 1) *
sizeof(int));
		}
		else {
			i++;
		}
	}
}
int main() {
	setlocale(0, "");

	int n;
	cin >> n;
	if (n < 1 || n > 1000) {
		cout << "Недопустимый размер массива";
		return 1;
	}

	int* massiv;
	massiv = (int*)malloc(n * sizeof(int));

	fill_in(massiv);

	int deleted_x_n2;
	cin >> deleted_x_n2;
	delete_all_comers_n2(&massiv, deleted_x_n2);

	cout << endl;
	print_massiv(massiv);

	return 0;
}
```

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 8 Массив: 5 8 9 4 6 2 1 7 Удаляемое значение: 2	5 8 9 4 6 1 7	5 8 9 4 6 1 7
2	Кол-во элементов: 7 Массив: 5 8 7 5 5 6 5 Удаляемое значение: 5	8 7 6	8 7 6
3	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Удаляемое значение: 11	1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10

2.4. Реализация операции удаления со сжатием всех вхождений заданного значения из динамического массива, модифицируя соответствующую функцию статического массива. Сложность алгоритма O(n).

```
void delete_all_comers_n(int** list, int x) {
	int j = 0;
	for (int i = 0; i < _msize(*list) / sizeof(int); i++) {
		(*list)[j] = (*list)[i];
		if ((*list)[i] != x) {
			j++;
		}
	}
	*list = (int*)realloc(*list, j * sizeof(int));
}

int main() {
	setlocale(0, "");

	int n;
	cin >> n;
	if (n < 1 || n > 1000) {
		cout << "Недопустимый размер массива";
		return 1;
	}

	int* massiv;
	massiv = (int*)malloc(n * sizeof(int));

	fill_in(massiv);

	int deleted_x_n;
	cin >> deleted_x_n;
	delete_all_comers_n(&massiv, deleted_x_n);

	cout << endl;
	print_massiv(massiv);

	return 0;
}
```

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 8 Массив: 5 8 9 4 6 2 1 7 Удаляемое значение: 2	5 8 9 4 6 1 7	5 8 9 4 6 1 7
2	Кол-во элементов: 7 Массив: 5 8 7 5 5 6 5 Удаляемое значение: 5	8 7 6	8 7 6
3	Кол-во элементов: 10 Массив: 1 2 3 4 5 6 7 8 9 10 Удаляемое значение: 11	1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10

ВЫВОДЫ

Функции должны принимать входные данные через параметры и возвращать результат, если этого требует алгоритм операции.

Приобретены практические навыки по работе с:

1. Статическими массивами;
2. Динамическими массивами (модулем malloc).

Разработаны операции для управления статическим массивом и реализованы функциями.

Реализованы операции вставки и удаления значений в динамическом массиве, модифицируя соответствующие функции статического массива. Остальные функции, разработанные для статического массива, дополнительных изменений не требуют.

Тестирования всех операций пройдены успешно.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Процедурное программирование Языки программирования – Сайт lizochekk! [Электронный ресурс]: URL: https://lizochekk.jimdofree.com/программирование/
2. Документация по Microsoft C/C++ | Microsoft Docs – [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/cpp/?view=msvc-160
3. C++ – Типизированный язык программирования / Хабр – [Электронный ресурс] URL: https://habr.com/ru/hub/cpp/