МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Пензенский государственный университет»

Кафедра «Математическое обеспечение и применение ЭВМ»

ОТЧЕТ

по учебной (ознакомительной) практике

Выполнил: Макаричева Е. М.

Направление: 09.03.04 «Программная инженерия»

Группа: 20ВП1

Руководитель практики: Самуйлов С. В.

Работа защищена

Оценка

Пенза, 2021

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Математическое обеспечение и применение ЭВМ»

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на учебную (ознакомительную) практику

Задание 1

1 Исследовать современные технологии объектно-ориентированного программирования на языке C++ в среде разработки Visual Studio.

2. Разработать приложение для следующего задания:

Дан целочисленный массив размера N. Увеличить все нечетные числа, содержащиеся в массиве, на исходное значение последнего нечетного числа. Если нечетные числа в массиве отсутствуют, то оставить массив без изменений.

Вывести исходные данные и результат работы программы

2.1. Разработать алгоритм решения задачи (диаграмма деятельности).

2.2. Выполнить тестирование разработанного приложения. Результаты представить в виде таблицы и скриншотов.

2.4. Привести структуру приложения (через обозреватель решений)

2.5. Представить код приложения.

3. Создать репозиторий в GitLab («Открытые решения») и поместить в него код приложения решения заданной задачи (п.2)

4. Представить сертификат и результаты курса (C++ или UML)

Задание 2

Изучить информационную технологию обработки текстовой информации и технической документации. Сделать обзор ключевых возможностей и особенностей применения данной технологии в различных программных средах. Изучить стандарт ГОСТ 2.105–95 «Общие требования к текстовым документам». Оформить отчет по практике в соответствии с ГОСТ 2.105–95 «Общие требования к текстовым документам».

Срок выполнения индивидуального задания 13 июля 2021 г.

Дата выдачи задания «29» июня 2021 г.

Дата защиты отчета «____» _____

Руководитель _____ С.В. Самуйлов

Задание получил «29» июня 2021 г.

Студент _____ Макаричева Е.М.

Оглавление

Введение

Во время прохождения учебной практики необходимо решить следующие задачи: проанализировать требования к структуре данных (в данном случае структура данных представляет собой массив), составить диаграмму вариантов использования в процессе анализа функциональных требований, разработать диаграмму деятельности, реализовать и протестировать программу.

Для моделирования программных средств будет использоваться язык UML.

UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем. Диаграммы UML предоставляют возможность разработчику посмотреть на задачу с разных точек зрения, а другим программистам будет легче понять суть задачи и способ ее реализации, так как диаграммы сравнительно просты для чтения после достаточно быстрого ознакомления с их синтаксисом.

Разработка будет осуществляться на языке C++ на платформе Microsoft Visual Studio.

Visual Studio - это интегрированная среда разработки (IDE) от Microsoft. VS включает в себя редактор кода, поддерживающий IntelliSense. Интеллектуальная система среды разработки умеет подстраивать код под необходимый формат и синтаксис. Таким образом, он становится более читаемым и доступным для редактирования.

Предлагаемые в Visual Studio инструменты отладки являются наилучшим средством для отслеживания ошибок и диагностирования неожиданного поведения работы программы. Разработчик может выполнять свой код по строке за раз, устанавливать интеллектуальные точки

прерывания, при желании сохраняя их для использования в будущем, и в любое время просматривать текущую информацию из памяти.

VS поддерживает различные языки программирования и позволяет редактору кода и отладчику поддерживать практически любой язык программирования при условии, что существует специальная языковая служба. C++, наряду с C# и JavaScript, является встроенным языком среды разработки.

1. Анализ требований

1.1. Требования к структуре данных

Массив – это конечная именованная последовательность однотипных величин.

Другими словами, массив – это тип или структура данных в виде набора компонентов (элементов массива), расположенных в памяти непосредственно друг за другом[1].

Описание массива в программе отличается от описания простой переменной наличием после имени квадратных скобок, в которых задается количество элементов массива.

int a [10]; // описание массива из 10 целых чисел

Доступ к элементам массива, осуществляется через индексацию. Элементы массива нумеруются с нуля. Инициализирующие значения для массивов записываются в фигурных скобках. Если в массиве элементов больше, чем инициализаторов, элементы, для которых значения не указаны, обнуляются.

int b[5] = {3, 2, 1}; // b[0] = 3, b[1] = 2, b[2] = 1; b[3] = 0; b[4] = 0

Размерность массива вместе с типом его элементов определяет объем памяти, необходимый для размещения массива, которое выполняется на этапе компиляции, поэтому размерность может быть задана только целой положительной константой или константным выражением[2].

1.2. Функциональные требования

Чтобы описать функциональное назначение системы или то, что система должна делать, воспользуемся диаграммой вариантов использования в UML.

Суть диаграммы вариантов использования состоит в следующем. Проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью вариантов использования. При этом актером называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое

устройство, программа или любая другая ситема, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.

Вариант использования служит для описания сервисов, которые система предоставляет актеру.

Диаграмма вариантов использования, состоит из четырех компонентов:

— контекст системы – прямоугольник, очерчивающий варианты использования для обозначения границ моделируемой системы;

— актера – это роль объекта вне системы, который напрямую взаимодействует с ее составными частями. Стандартным графическим обозначением актера на диаграммах является фигурка человечка, под которой записывается имя актера;

— вариантов использования – то, что актеры могут делать с системой. Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его описание, обозначающее выполнение какой-либо операции или действия;

— отношений – значимые отношения между элементами моделируемой системы.

Связи между актерами и вариантами использования отображаются с использованием четырех видов отношений:

— Отношение ассоциации служит для обозначения взаимодействия актера с вариантом использования. Оно обозначается сплошной линией между актером и вариантом использования. Эта линия может иметь условные обозначения, такие как имя и кратность. Отношение ассоциации Кратность ассоциации указывается рядом с обозначением компонента диаграммы, который является участником данной ассоциации, и характеризует количество экземпляров данного компонента, которые могут выступать в качестве элементов данной ассоциации. Применительно к

диаграммам вариантов использования кратность имеет специальное обозначение в форме одной или нескольких цифр и символа «звездочка». Если кратность отношения ассоциации не указана, то, по умолчанию, принимается ее значение, равное единице;

— Отношение обобщения служит для указания того факта, что некоторая сущность А может быть обобщена до сущности В. В этом случае сущность А будет являться специализацией сущности В. На диаграмме данный вид отношения можно отображать только между однотипными сущностями (между двумя вариантами использования или двумя актерами). Графически данное отношение обозначается сплошной линией со стрелкой, в виде не закрашенного треугольника, от потомка к родителю;

— Отношение расширения определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров. В метамодели отношение расширения является направленным и указывает, что применительно к отдельным примерам некоторого варианта использования должны быть выполнены конкретные условия, определенные для расширения данного варианта использования. Отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от того варианта использования, который является расширением для исходного варианта использования. Данная линия со стрелкой помечается ключевым словом «extend»;

— Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования.

Графически данное отношение обозначается пунктирной линией со стрелкой, которая помечается ключевым словом «include» (включает). Стрелка включения должна быть направлена от базового (составного) варианта к включаемому.

Для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта, в языке UML предназначены примечания. Графически примечания обозначаются прямоугольником с загнутым верхним правым углом. Внутри прямоугольника содержится текст самого примечания[3].

В процессе анализа функциональных требований была составлена диаграмма вариантов использования.

Диаграмма вариантов использования, разработанная с помощью средства UML моделирования – AgroUML, приведена на рисунке 1.

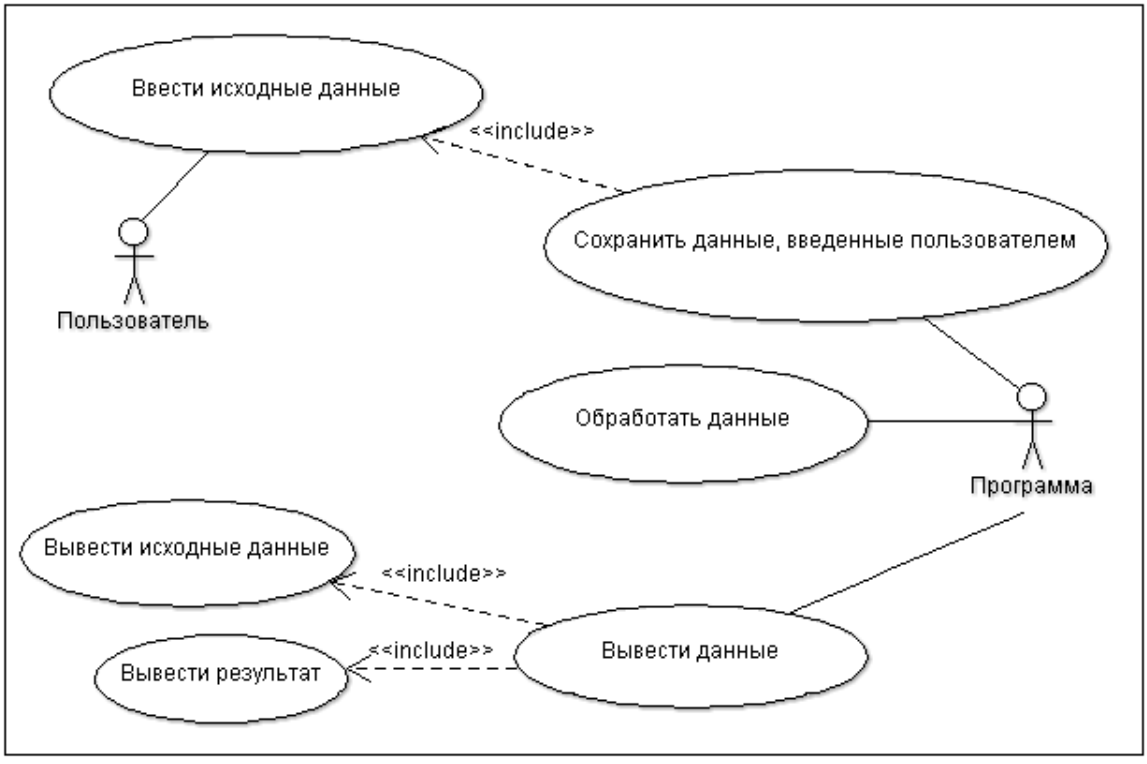

Рисунок 1 – Диаграмма вариантов использования

На диаграмме вариантов использования выделены следующие актеры: пользователь и программа. Пользователь имеет один вариант использования:

«Ввести исходные данные», они связаны отношением ассоциации. Актер Программа имеет три варианта использования. «Сохранить данные, введенные пользователем», «Обработать данные» «Вывести данные». Вариант использования «Сохранить данные, введенные пользователем» связан отношением включения с вариантом использования «Ввести исходные данные» пользователя, так как для сохранения данных программа сначала должна их получить. Варианты использования «Вывести исходные данные» и «Вывести результат» являются составными компонентами варианта использования «Вывести данные», поэтому они связаны между собой отношением включения.

2. Разработка приложения

На начальном этапе разработки приложения возникает необходимость в детализации алгоритмической реализации операций, выполняемых системой[4].

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций, когда необходимо представить алгоритмы их выполнения. В контексте языка UML деятельность представляет собой совокупность отдельных вычислений, выполняемых объектом, приводящих к некоторому результату или действию. На диаграмме деятельности отображаются логика и последовательность переходов от одной деятельности к другой, а внимание аналитика фокусируется на результатах. Результат деятельности может привести к изменению состояния системы или возвращению некоторого значения.

Компоненты диаграммы деятельности:

Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий (псевдосостояние). В этом состоянии находится объект по умолчанию в начальный момент времени. Оно служит для указания на диаграмме графической области, от которой начинается процесс изменения состояний. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка, из которого может только выходить, минимум, одна стрелка, соответствующая переходу.

Конечное состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий (псевдосостояние). В этом состоянии объект будет находиться непосредственно перед уничтожением после завершения потока управления. Графически конечное состояние в языке UML обозначается в виде закрашенного кружка, помещенного в окружность, в которую может входить, минимум, одна

стрелка, соответствующая переходу. Конечное состояние служит для указания на диаграмме графической области, в которой завершается процесс изменения состояний или жизненный цикл данного объекта.

Состояние действия является специальным случаем состояния с некоторым входным действием и, по крайней мере, одним выходящим из состояния переходом. Этот переход неявно предполагает, что входное действие уже завершилось. Состояние действия не может иметь внутренних переходов, поскольку оно является элементарным. Обычное использование состояния действия заключается в моделировании одного шага выполнения алгоритма (процедуры) или потока управления. Графически состояние действия изображается прямоугольником с закругленными углами. Любое состояние действия должно иметь, минимум, одну стрелку входящего перехода и, минимум, одну стрелку, соответствующую исходящему переходу. Внутри этого изображения записывается выражение действия, которое должно быть уникальным в пределах одной диаграммы деятельности.

Простой переход представляет собой отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния объекта другим. На диаграмме деятельности переход изображается сплошной линией со стрелкой, которая направлена в целевое состояние. Если пребывание моделируемого объекта в первом состоянии сопровождается выполнением некоторых действий, то переход во второе состояние будет возможен только после завершения этих действий и, возможно, после выполнения некоторых дополнительных условий, называемых сторожевыми условиями.

Событие является самостоятельным элементом языка UML. В качестве событий можно рассматривать сигналы, вызовы, окончание фиксированных промежутков времени или моменты окончания выполнения определенных действий. Имя события идентифицирует каждый отдельный переход на диаграмме деятельности и может содержать строку текста, начинающуюся со

строчной буквы. Если переход сопровождается событием, то его принято триггерным, т.е. таким, который специфицирует событие-триггер. Если рядом со стрелкой перехода не указана никакая строка текста, то соответствующий переход является не триггерным, и в этом случае из контекста диаграммы деятельности должно следовать, после окончания какой деятельности он выполняется. После имени события могут следовать круглые скобки для явного задания параметров соответствующего события-триггера. В общем случае список параметров содержит целый ряд отдельных параметров, разделенных символом «,». Если таких параметров нет, то список параметров со скобками может отсутствовать.

Сторожевое условие, если оно есть, всегда записывается в прямых скобках после события-триггера и представляет собой некоторое булевское (логическое) выражение.

Графически ветвление на диаграмме деятельности обозначается небольшим ромбом, внутри которого нет никакого текста. В этот ромб может входить только одна стрелка от того состояния действия, после выполнения которого поток управления должен быть продолжен по одной из взаимно исключающих ветвей. Принято входящую стрелку присоединять к верхней или левой вершине символа ветвления. Выходящих переходов может быть два или более, но для каждого из них явно указывается соответствующее сторожевое условие в форме логического[3].

В ходе прохождения учебной практики была решена задача увеличения всех нечетных чисел массива, на исходное значение последнего нечетного числа.

Алгоритм решения поставленной задачи представлен в виде диаграммы деятельности, изображенной на рисунке 2. Диаграмма деятельности создана с помощью средства UML моделирования – AgroUML.

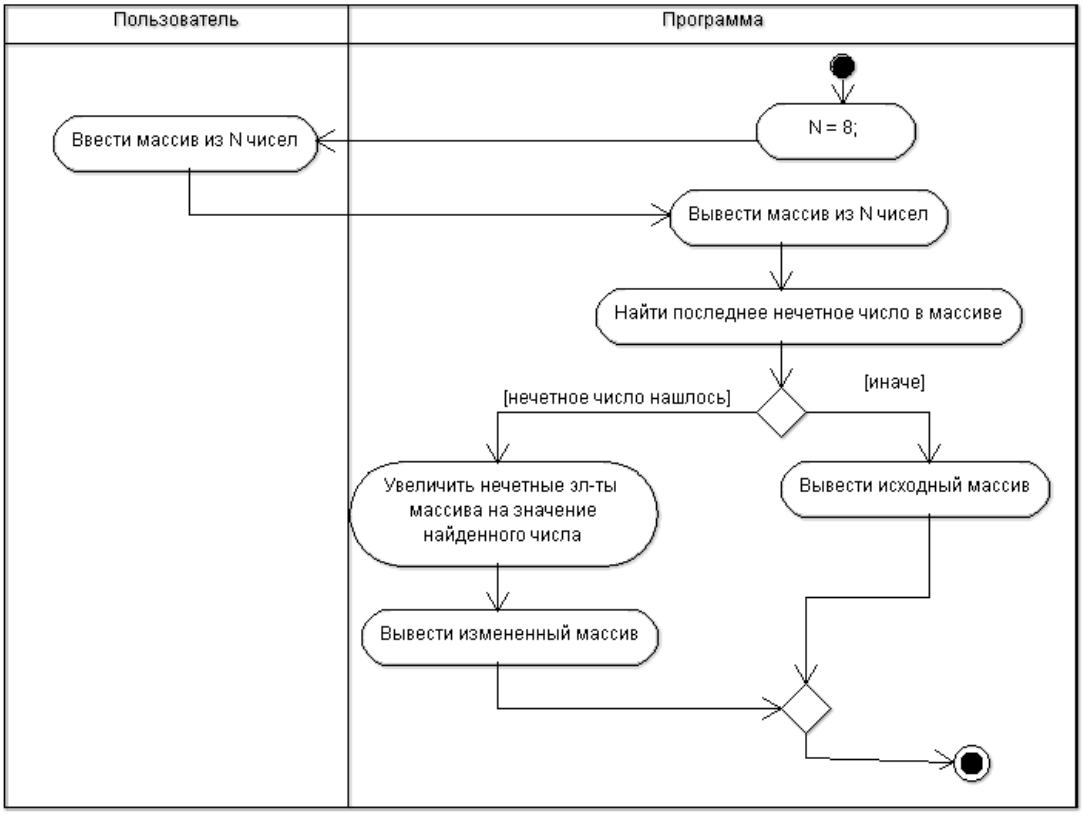

Рисунок 2 – Диаграмма деятельности

На диаграмме деятельности представлены две дорожки: Пользователь и Программа. Пока пользователь не введет исходные данные, программа работать не сможет. Так как по заданию необходимо увеличить нечетные элементы массива на значение последнего нечетного числа, то, прежде всего, нужно найти последний нечетный элемент массива, то есть элемент с наибольшим индексом. Так как нечетных чисел в массиве может и не быть, то программа должна вывести исходный массив значений. Если же такие числа в массиве есть и последнее нечетное число найдено, то программа увеличивает все нечетные элементы массива на найденное значение. В результате выводится массив с измененными значениями.

Реализация программы

2.1. Кодирование

В результате прохождения практики была разработана программа решения поставленной задачи. Задача состояла в следующем: необходимо было увеличить все нечетные числа, содержащиеся в массиве, на исходное значение последнего нечетного числа. Если же в массиве нет нечетных чисел, то необходимо было вывести исходный массив. Код программы приведен в Приложении А.

2.2. Тестирование

В результате выполнения задания было выполнено функциональное тестирование разработанной программы.

Таблица тестирования приведена ниже.

Таблица 1 – Таблица тестирования

Состав теста	Ожидаемый результат	Наблюдаемый результат
Тест 1 1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8 8 2 10 4 12 6 14 8	1 2 3 4 5 6 7 8 8 2 10 4 12 6 14 8 (см. Рисунок 3)
Тест 2 2 4 6 8 6 4 2 10	2 4 6 8 6 4 2 10 2 4 6 8 6 4 2 10	2 4 6 8 6 4 2 10 2 4 6 8 6 4 2 10 (см. Рисунок 7)
Тест 3 2 -1 3 8 6 -2 -3 4	2 -1 3 8 6 -2 -3 4 2 -4 0 8 6 -2 -6 4	2 -1 3 8 6 -2 -3 4 2 -4 0 8 6 -2 -6 4 (см. Рисунок 8)
Тест 4 2 5 g 1 4 9 6 4	Сообщение с требованием изменения 3 элемента массива, до тех пор, пока не будет введено число, затем возможность продолжения ввода	Сообщение «Неверный пользовательский ввод! Введите элемент:» После замены элемента верным значением появилось сообщение: «Введите 4 элемент:» (см. Рисунок 4)

Продолжение Таблицы 1

Состав теста	Ожидаемый результат	Наблюдаемый результат
Тест 5 /	Сообщение с требованием изменения элемента, до тех пор, пока не будет введено число, затем возможность продолжения ввода	Сообщение «Неверный пользовательский ввод! Введите элемент:» После замены элемента верным значением появилось сообщение: «Введите 2 элемент:» (см. Рисунок 5)
Тест 6 0 0 0 0 0 0 0 a	При вводе 8 элемента выдается сообщение с требованием изменения 8 элемента массива	Сообщение «Неверный пользовательский ввод! Введите элемент:» (см. Рисунок 6)

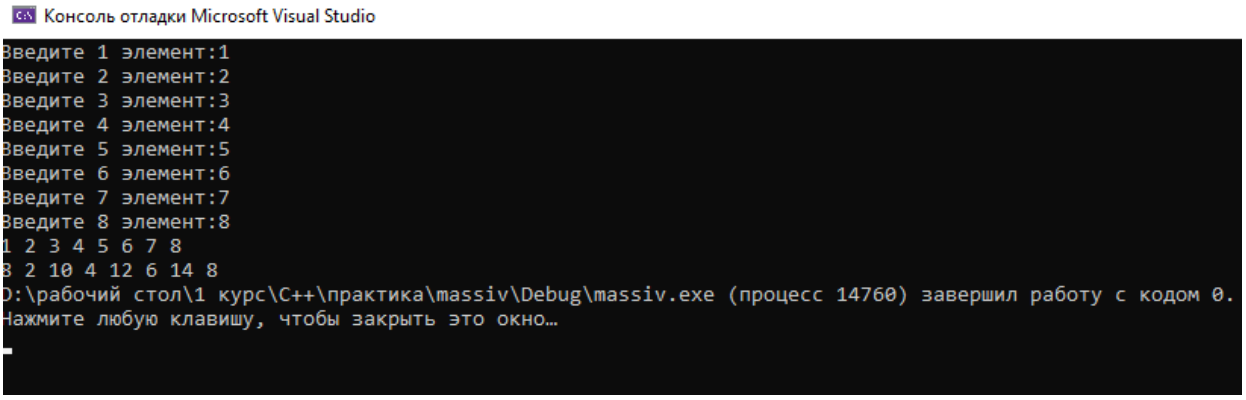

Рисунок 3 – Тест 1

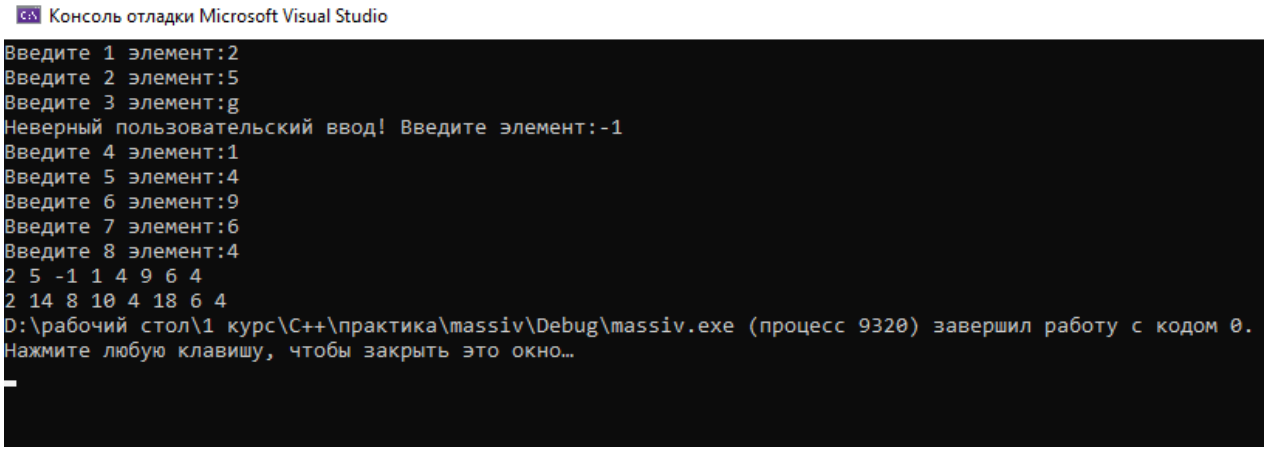

Рисунок 4 – Тест 4

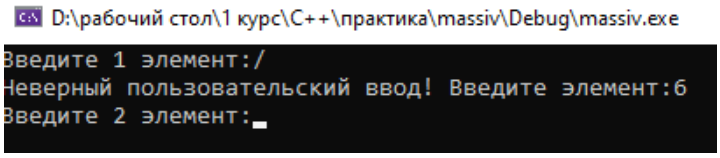

Рисунок 5 – Тест 5

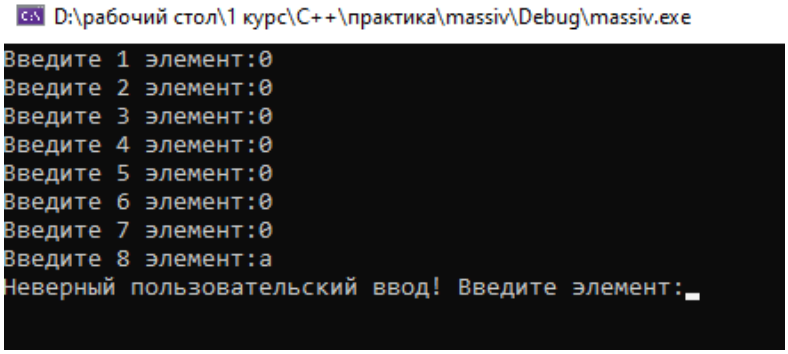

Рисунок 6 – Тест 6

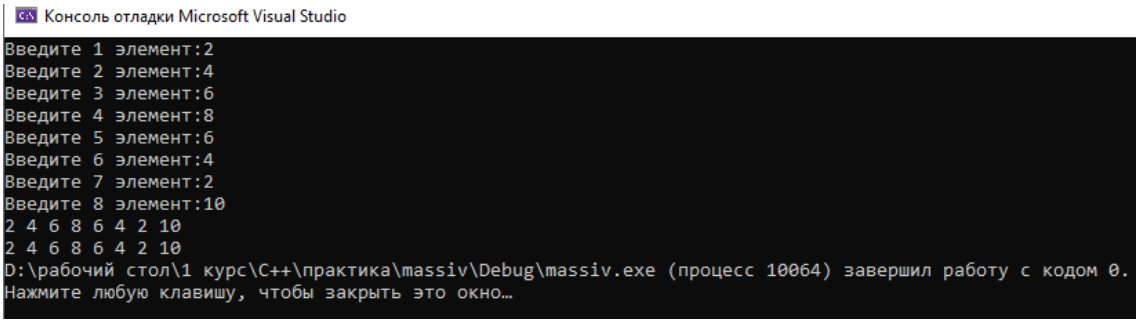

Рисунок 7 – Тест 2

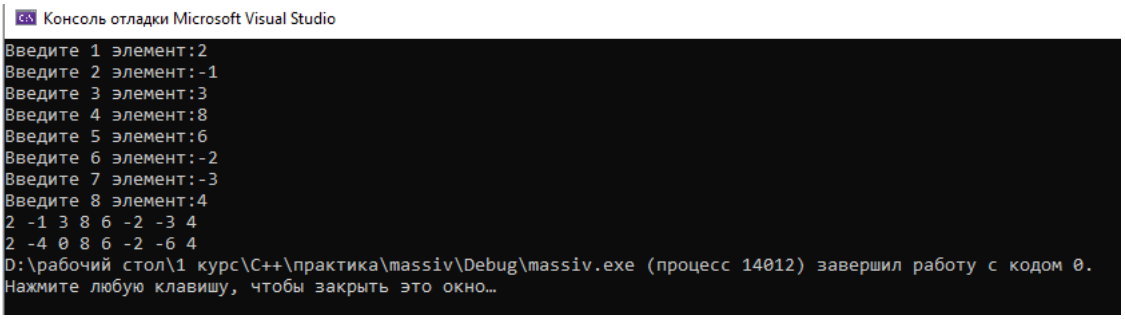

Рисунок 8 – Тест 3

В ходе тестирования разработанной программы были введены как верные исходные данные, так и ошибочные. Все тесты выдали ожидаемые результаты. Тестирование пройдено успешно.

Структура разработанного приложения под названием «massiv» приведена на рисунке 9.

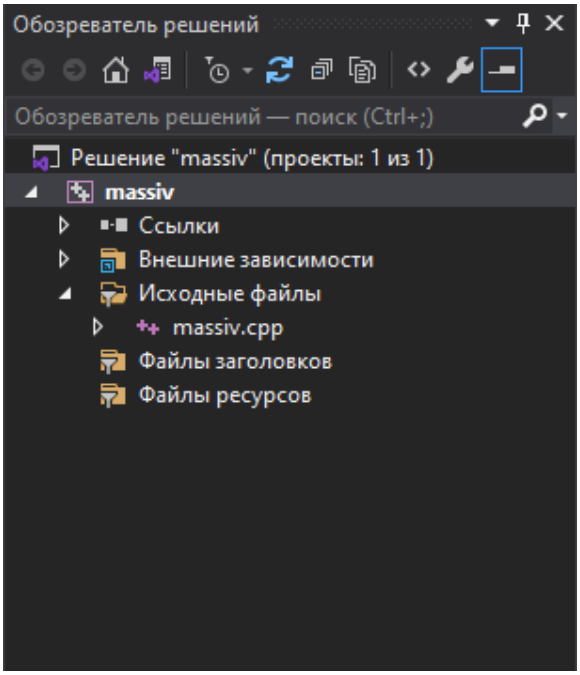

Рисунок 9 – Структура приложения

Заключение

В процессе прохождения учебной практики были выполнены следующие задачи:

- проанализированы требования к структуре данных;
- рассмотрено функциональное назначение программы с помощью построения и анализа диаграммы вариантов использования UML;
- разработан алгоритм решения поставленной задачи, благодаря диаграмме деятельности, построенной с помощью средства UML моделирования – AgroUML;
- написан и протестирован код разработанного приложения; Результаты тестирования представлены в виде таблицы и скриншотов;

Также в период прохождения практики был получен сертификат Национального Открытого Университета «ИНТУИТ» по курсу «Алгоритмизация. Введение в язык программирования C++». Сертификат приведен в приложении Б.

Список использованных источников

1. Массив (тип данных): сайт. –
   URL: https://wiki2.org/ru/Массив_(тип_данных) (дата обращения: 30.06.2021)
2. Т. А. Павловская. C/C++. Программирование на языке высокого уровня. – СПб.: Питер, 2003. – 461 с.: ил.
3. Балашова И. Ю. Современные информационные технологии в проектировании программных систем и комплексов : учеб. пособие / И. Ю. Балашова, Д. В. Такташкин ; под ред. П. П. Макарычева. – Пенза : Изд-во ПГУ, 2019. – 106 с.
4. Диаграмма деятельности(activity diagram): сайт. –

URL: http://www.telenir.net/uchebniki/samouchitel_uml/p7.php (дата обращения 1.07.2021)

Приложение А.

ЛИСТИНГ ПРОГРАММЫ

```cpp
#include <iostream>
using namespace std;

int main()
{
	setlocale(LC_ALL, "Russian");
	const int N = 8;
	int a[N];
	int i = 0;
	cout << "Введите 1 элемент:";
	while (i < N) {
		cin >> a[i];
		if (!cin) {
			cout << "Неверный пользовательский ввод! Введите
				элемент:";
			cin.clear(); // восстановление работы потока
			cin.get();// ожидание ввода пользователя
			continue;
		}
		i++;
		if(i < N)
			cout << "Введите " << i + 1 << " элемент:";
	}
	for (int i = 0; i < N; i++) {
		cout << a[i]<< " ";
	}
	cout << endl;
	int nech = 0;
	for (int i = 0; i < N; i++) {
		if (a[i] % 2 != 0) {
			nech = a[i];
		}
	}
	//cout << "nech: " << nech << endl;
	if (nech == 0) {
		//cout << "Нечетных чисел в массиве нет!";
		for (int i = 0; i < N; i++) {
			cout << a[i] << " ";
		}
	}
	else {
		for (int i = 0; i < N; i++) {
			if (a[i] % 2 != 0) {
				a[i] += nech;
			}
			cout << a[i]<<" ";
		}
	}
}
```

Приложение Б.

РЕЗУЛЬТАТЫ ПРОХОЖДЕНИЯ ОБУЧАЮЩЕГО КУРСА

ИНТУИТ
НАЦИОНАЛЬНЫЙ ОТКРЫТЫЙ УНИВЕРСИТЕТ

Сертификат

Серия О **Регистрационный № 101467057**

ВЫДАН

Макаричевой Елизавете Михайловне

в том, что она с 26 июня 2021 по 10 июля 2021 прошла обучение в Национальном Открытом Университете «ИНТУИТ» по курсу «Алгоритмизация. Введение в язык программирования C++» в объеме 72 часов

Ректор Герасина

г. Москва 10 июля 2021 года