Балаковский инженерно-технологический институт - филиал

федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ»

Факультет атомной энергетики и технологий

Кафедра «Информатика и управление в технических системах»

КОНТРОЛЬНАЯ РАБОТА

по дисциплине

«Название дисциплины»

Выполнил: студент группы ИФСТ-1з

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Семенов М.А.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_\_г.

Проверил ст. преподаватель каф. ИУС

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Михеев И.В.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_\_г.

Балаково 2022

Введение

Пользовательский интерфейс — это все, что помогает людям управлять устройствами и программами с помощью голоса, нажатий, жестов, через командную строку и даже силой мысли (такое теперь тоже есть). Самый популярный вид интерфейсов сейчас — UI приложений.

UI (англ. user interface) переводится как «пользовательский интерфейс». UI охватывает не только графический интерфейс, а еще и тактильный, голосовой или звуковой.

Интерфейс помогает двум объектам понимать друг друга и обмениваться информацией.

Интерфейс — это «язык общения», который понимают оба объекта, которые взаимодействуют друг с другом с целью решить определенный вопрос.

Если каждое приложение или программа, установленная на компьютере, планшете или смартфоне, — это помощник, то интерфейс — это способ общаться (взаимодействовать) с ней, чтобы она помогала в вашем деле на работе и в жизни.

Также распространены программный, аппаратный, аппаратно-программный интерфейсы. Такие интерфейсы обеспечивают взаимодействие не только между человеком и машиной (устройством), но и между программами, оборудованием или компьютерами:

* + аппаратный: соединяет друг с другом два объекта, например, помогает подключить смартфон к ноутбуку с помощью WiFi или кабеля;
  + программный (API): создает связь между приложениями/программами, к примеру, подключение API одного приложения к другому. Самый популярный сценарий работы — авторизация через соцсети на сайтах;
  + аппаратно-программный: комбинация технических элементов под управлением программного обеспечения.

1. Виды пользовательского интерфейса

Пользовательские интерфейсы бывают жестовые, тактильные, голосовые, графические, командной строки и даже нейронные.

Интерфейс командной строки и текстовый интерфейс (Command Line Interface или CLI)

Командная строка все еще очень популярна среди системных администраторов и программистов. Это один из первых методов взаимодействия с компьютером. Она обладает особым шармом — создает ощущение общения тет-а-тет с машиной без посредников. Командная строка — как бесконечный лист A4, на котором пользователь вводит текст команд и получает результаты работы в виде текста.

Графический пользовательский интерфейс (Graphical User Interface или GUI)

Самый популярный тип UI. Представляет собой окошко с различными элементами управления. Пользователи взаимодействуют с ними с помощью клавиатуры, мыши и голосовых команд: жмут на кнопки, тыкают мышкой, смахивают пальцем.

Жестовый, голосовой, тактильный, нейронный «Любая достаточно развитая технология неотличима от магии», — как-то сказал английский писатель-фантаст и футуролог Артур Кларк.

Например, через Voice User Interface вы можете отдавать команды своему смартфону через голосовых помощников: Siri от Apple, Alexa от Amazon или Алиса от Яндекса.

NUI (жестовые, естественные) применяют в играх для приставок Xbox, Nintendo Wii или PlayStation. Эту же технологию вы найдете в оборудовании «умного дома», например, при включении света или регулировании громкости Яндекс.Станции с помощью изменения положения руки.

Производители качают технологии и расширяют возможности машин, и наслаждаться новыми фишками гаджетов можно даже посылая мысли напрямую в компьютер.

Графический пользовательский интерфейс

Этим термином чаще обозначаются UI мобильных и веб-приложений, а также игр и сервисов для развлечений.

Мобильные интерфейсы

Выделяется в отдельную группу SIMP (Screen, Icon, Menu, Pointer). Подход к дизайну мобильных интерфейсов отличается от подхода к дизайну настольных приложений. Поведение пользователей при взаимодействии со смартфонами отличается от работы на компьютере из-за размера экрана и отсутствия отдельной клавиатуры с мышью/тачпадом. Элементы здесь заполняют экран полностью, а блоки и системы зависят от требований операционной системы.

Дизайн мобильных приложений также зависит от поведенческих паттернов пользователей, например того, как они держат смартфон в руке, какие действия удобно совершать на ходу и т.д.

Веб-интерфейсы

Технологии позволяет создавать полноценные веб-приложения, по функциональности не уступающие настольному ПО: Trello, Google Docs, Twitch, Яндекс.Дзен.

Преимущество таких приложений в том, что их не нужно устанавливать на компьютер — все функции доступны в браузере. Создают такие приложения с помощью JavaScript, HTML и CSS.

Игровой и материальный

Связан с механикой геймплея. Именно в нем лучше всего раскрывается сопровождающая роль интерфейса, так как игрок лучше ощущает, что вижется к какой-то цели (например, победить босса и пройти уровень). Интерфейс зависит от игры: кнопки, жесты, движения мыши или взаимодействие с сенсором на экране или 3D интерфейс в VR, нажатие клавиш на джойстике.

Модель пользовательского интерфейса: реальный мир и ментальная модель пользователя

Программные продукты призваны увеличивать наши возможности в реальном мире. Каждый продукт — как супергерой, его задача — помочь нам в чем-либо: суперпамять, общение сквозь любые расстояния, максимум развлечений и так далее. Ко всем этим свойствам мы получаем доступ через интерфейсы.

Каждое приложение мы распахиваем в определенном контексте. Контекст подразумевает определенные ожидания от того, как все должно работать. Ожидания основаны на прошлом опыте. При знакомстве с новым продуктом мы бессознательно переносим на него сформировавшиеся ожидания и привычки, которые выстроились в прошлом вокруг другого схожего продукта (или способа решения схожей задачи).

У каждого интерфейса под капотом находится определенная модель системы, которая призвана помогать пользователю достигать определенных целей.

Например, модель покупки дорогих автомобилей для постоянных клиентов автосалона в мобильном приложении может включать 4 шага: наполнить корзину, оформить заказ, подтвердить заказ, внести оплату.

Модель и Интерфейс тесно взаимосвязаны. Чтобы понять разницу между Моделью и Интерфейсом, задайте вопрос: откуда вообще взялась корзина? Разве клиенты добавляют автомобили в тележку пачками, как в супермаркете? Или все-таки нужна модель не корзины/тележки из супермаркета, а тест-драйва в автосалоне?

Т.е. мы в приложении можем вместо корзины сразу при выборе автомобиля предлагать записаться на тест-драйв или начать оформление покупки (в том числе в кредит). А теперь задайте вопрос: как часто покупатели дорогого автомобиля готовы менять авто и вписывается ли мобильное приложение в их жизненный контекст?

Выходит, что за любым объектом в реальном мире стоит модель, которая находится в нашем сознании — это наши представления о том, как эта штуковина перед нами должна работать. И эти представления возникают еще до того, как мы дотронулись до нее.

На рисунке 1 изображены типы интерфейсов.

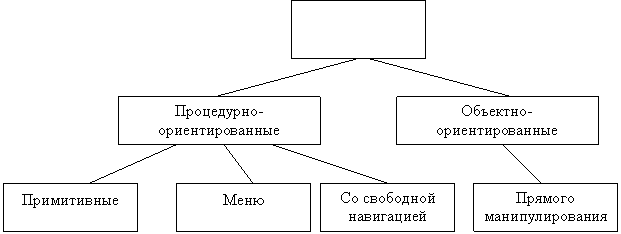


Рисунок 1 типы интерфейсов

1. Этапы разработки пользовательского интерфейса — как проработать UI

В международной практике подход к дизайну интерфейсов уже стал стандартом. Процесс по дизайну интерфейсов включает следующие ключевые этапы. В этом блоке я опираюсь на материалы UX Mastery — партнера Interaction Design Foundation, крупнейшего в мире сообщества обучения UX-дизайну.

* Стратегия (Бренд-стратегия и UX-стратегия) — определяет полезное действие, ценности бренда и видение будущего. Стратегия естественным образом влияет на цели проекта по дизайну интерфейсов, критерии достижения целей и приоритет проекта в общем ландшафте высот организации.
* Исследование (UX-исследование) — фаза открытий. Комплексные проекты включают в себя масштабную работу по пользовательским исследованиям (UX-исследованиям) и анализу конкурентов (бенчмаркинг). Небольшие организации или стартапы могут подойти к исследовательской работе в упрощенном формате и обосновать идею, построенную по принципам минимальной жизнеспособности (Minimum Viable) через интервью, опросы и юзабилити тестирования. Именно на фазе исследований, согласно принципам дизайн-мышления, происходит погружение в образ жизни, пристрастия, цели и барьеры пользователей. Понимание контекста пользователей помогает создавать социально значимые продукты, которые с высокой вероятностью приживутся и начнут развитие на рынке: так они будут казаться интуитивными (ментальные модели) и родными (метафоры).
* Анализ (UX-аналитика) — цель анализа в том, чтобы сделать выводы из данных и дать уверенный старт созданию дизайн-концепций. Выводы призваны помочь понять суть происходящего и приступить к проектированию интерфейса.
* Проектирование и прототипирование интерфейсов — на этапе проектирования происходит создание прототипов интерфейсов, их тестирование пользователями и корректировка на основе обратной связи. На этой фазе чаще применяются прототипы с низкой детализацией (Low-fi prototyping), так в них пользователи фокусируются только на функциях и не отвлекаются на бренд-дизайн (уникальную графическую идентичность) и другие визуальные детали.
* Дизайн интерфейсов и Разработка — на этом этапе создается проработанный дизайн, пишется детальный контент, создается вся уникальная графика и начинается совместная работа с программистами.

Практическая работа

В данном задании необходимо провести анализ переменных, используемых в программе, проверить входные и выходные данные.



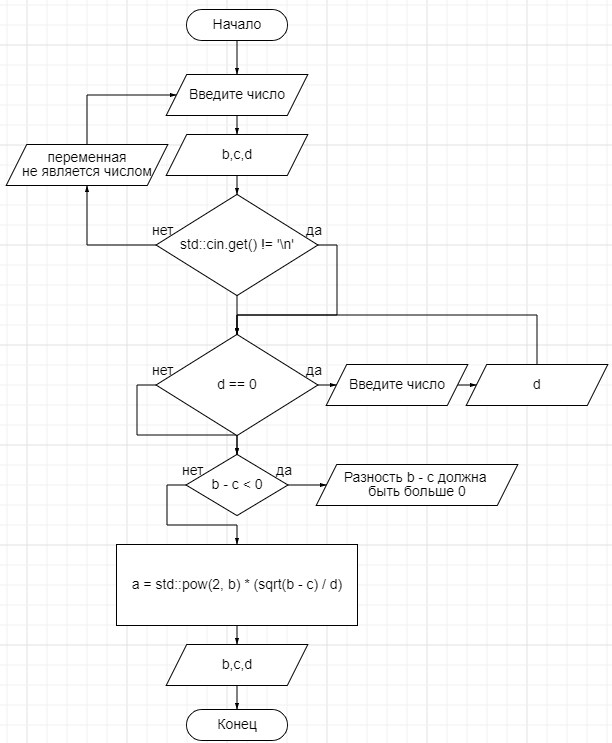


Рисунок 1 – Блок схема алгоритма

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Ожидаемый результат | Фактический результат | Результат тестирования |
| b=1,c=1,d=2 | 0 | 0 | Успешно |
| b=1,c=2,d=2 | Errors Разность b - c должна быть больше 0 | Errors Разность b - c должна быть больше 0 | Успешно |
| b=1,c=1,d=0 | Errors d должно быть не 0 и продолжится цикл ввода | Errors d должно быть не 0 и продолжится цикл ввода | Успешно |
| b=0.5,c=0.2,d=5 | 0.154919 | 0.154919 | Успешно |
| b=аф,c=0.2,d=5 | Errors b не является числом и продолжится цикл ввода | Errors b не является числом и продолжится цикл ввода | Успешно |

Исходный код программы.

#include <iostream>

#include <math.h>

#include <string>

float checkCin(std::string name) {

float res = 0;

std::cout << "Введите число " + name + "\n";

while (!(std::cin >> res) || (std::cin.peek() != '\n'))

{

std::cin.clear();

while (std::cin.get() != '\n');

std::cout << name + " является числом!\n";

std::cout << "Введите число" + name + "\n";

}

return res;

}

float checkZero(std::string name, float val) {

while (val == 0) {

std::cout << name + " должно быть не 0 \n";

std::cout << "Введите число " + name + "\n";

std::cin >> val;

}

return val;

}

int main(){

setlocale(LC\_CTYPE, "rus"); // вызов функции настройки локали

float a = 0;

float b = 0;

float c = 0;

float d = 0;

b = checkCin("b");

c = checkCin("c");

d = checkCin("d");

d = checkZero("d", d);

if (b - c < 0) {

std::cout << "Разность b - c должна быть больше 0";

}

else {

a = std::pow(2, b) \* (sqrt(b - c) / d);

std::cout << "итог " << a;

}

return 0;

}

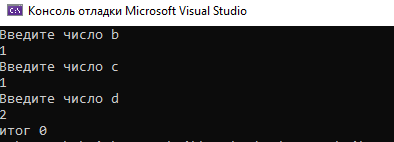


Рисунок 2 – Пример работы программы

В данном задании необходимо провести тестирование разработанного программного средства методом белого ящика.

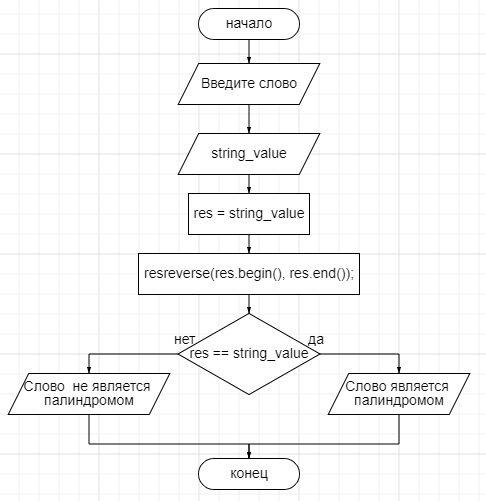


Рисунок 3 – Блок схема алгоритма

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Ожидаемый результат | Фактический результат | Результат тестирования |
| string\_value= привет | Слово не является  палиндромом | Слово не является  палиндромом | Успешно |
| string\_value= анина | Слово является  палиндромом | Слово является  палиндромом | Успешно |
| string\_value= кабак | Слово является  палиндромом | Слово является  палиндромом | Успешно |

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

int main() {

setlocale(LC\_CTYPE, "rus");

string string\_value;

cout << "Введите слово \n";

cin >> string\_value;

string res = string\_value;

reverse(res.begin(), res.end());

if (string\_value == res) {

cout << "Слово является палиндромом";

}

else {

cout << "Слово не является палиндромом";

}

return 0;

}

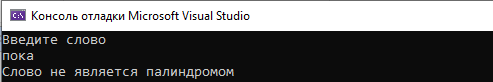


Рисунок 4 – Пример работы программы

Выводы

В работе были выполнены задания, которые научили проводить анализ переменных, используемых в программе, проверять входные и выходные данные.

Также проведено тестирование разработанного программного средства методом белого ящика. Тестирование по принципу белого ящика характеризуется степенью, какие тесты выполняют или покрывают логику.

Были спроектированы блок схемы программы, в которых показано какие валидации входных и выходных параметров должны обрабатываться программой.

На основе спроектированных блок схемы были разработаны программы на языке С++ и проведены тесты ожидаемый результатов этих программ.