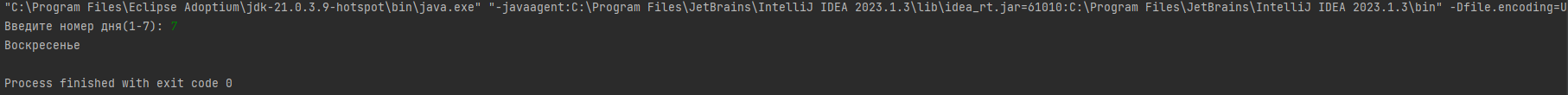
Практическая работа 4

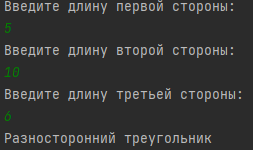
Определить день недели по номеру:

fun getDayOfWeek(dayNumber: Int):  
 String {return when(dayNumber){  
 1 -> "Понедельник"  
 2 -> "Вторник"  
 3 -> "Среда"  
 4 -> "Четверг"  
 5 -> "Пятница"  
 6 -> "Суббота"  
 7 -> "Воскресенье"  
 else -> "Некорректный номер дня"  
}}  
fun main() {  
 *print*("Введите номер дня(1-7): ")  
 val dayNumber = *readLine*()!!.*toInt*()  
 *println*(*getDayOfWeek*(dayNumber))  
}



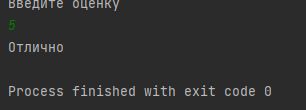
2. Определить тип треугольника по длинам сторон:

fun getTriangleType(a: Double, b: Double, c: Double):  
 String{return when{  
 a<=0||b<=0||c<=0 -> "Некорректные значения сторон"  
 a + b <= c || b+c <= a || a+c <= b -> "Треугольник не существует"  
 a==b && b==c -> "Равносторонний треугольник"  
 a==b || b==c || a==c -> "Равнобедренный треугольник"  
 a\*a + b\*b == c\*c || b\*b + c\*c == a\*a || a\*a + c\*c == b\*b -> "Прямоугольный треугольник"  
 else -> "Разносторонний треугольник"  
}}  
fun main() {  
 *println*("Введите длину первой стороны: ")  
 val a = *readLine*()!!.*toDouble*()  
 *println*("Введите длину второй стороны: ")  
 val b = *readLine*()!!.*toDouble*()  
 *println*("Введите длину третьей стороны: ")  
 val c = *readLine*()!!.*toDouble*()  
 if (a != null && b != null && c != null) {  
 *println*(*getTriangleType*(a, b, c))  
 } else {  
 *println*("Некорректные значения сторон")  
 }  
}



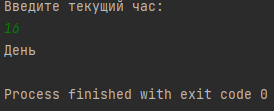
3. Вывод оценок по числовым значениям:

fun getGrade(score: Int): String {  
 return when(score) {  
 2 -> "Неудовлетворительно"  
 3 -> "Удовлетворительно"  
 4 -> "Хорошо"  
 5 -> "Отлично"  
 else -> "Некорректное значение"  
 }  
}  
fun main() {  
 *println*("Введите оценку")  
 val score = *readLine*()!!.*toInt*()  
 *println*(*getGrade*(score))  
}



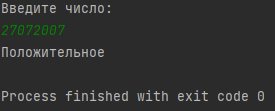
4. Определение времени суток:

fun getTimeOfDay(hour: Int): String {  
 return when (hour) {  
 in 0..5 -> "Ночь"  
 in 6..11 -> "Утро"  
 in 12..17 -> "День"  
 in 18..23 -> "Вечер"  
 else -> "Некорректное время"  
 }  
}  
fun main(){  
 *println*("Введите текущий час: ")  
 val hour = *readLine*()!!.*toInt*()  
 *println*(*getTimeOfDay*(hour))  
}



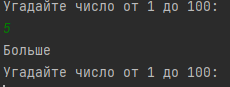
5. Определить знак числа:

fun getNumberSign(number: Int): String {  
 return when {  
 number > 0 -> "Положительное"  
 number < 0 -> "Отрицательное"  
 else -> "Ноль"  
 }  
}  
fun main(){  
 *println*("Введите число: ")  
 val number = *readLine*()!!.*toInt*()  
 *println*(*getNumberSign*(number))  
}



6. Угадай число

fun main() {  
 val number = (1..100).*random*()  
 while (true) {  
 *println*("Угадайте число от 1 до 100:")  
 val guess = *readLine*()?.*toIntOrNull*() ?: continue  
 when {  
 guess < number -> *println*("Больше")  
 guess > number -> *println*("Меньше")  
 else -> {  
 *println*("Угадали!")  
 break  
 }  
 }  
 }  
}



7. Определение длины строки

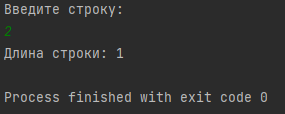
fun main() {

println("Введите строку:")

val text = readLine()

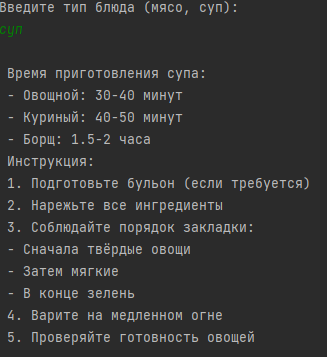
println("Длина строки: ${text?.length ?: 0}")

}



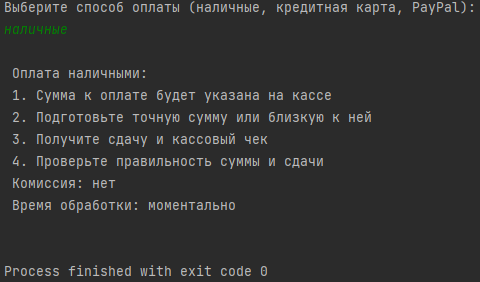
8. Определить время приготовления по типу блюда:

fun getCookingInstructions(foodType: String): String {  
 return when(foodType.*lowercase*()) {  
 "мясо" -> """  
 Время приготовления мяса:  
 - Стейк (средней прожарки): 4-5 минут с каждой стороны  
 - Свинина: 30-40 минут  
 - Говядина (тушение): 1.5-2 часа  
 Инструкция:  
 1. Достаньте мясо за час до готовки  
 2. Посолите и поперчите  
 3. Разогрейте сковороду/духовку  
 4. Для стейка: обжарьте на сильном огне  
 5. Для тушения: обжарьте, затем тушите на медленном огне  
 """  
  
 "суп" -> """  
 Время приготовления супа:  
 - Овощной: 30-40 минут  
 - Куриный: 40-50 минут  
 - Борщ: 1.5-2 часа  
 Инструкция:  
 1. Подготовьте бульон (если требуется)  
 2. Нарежьте все ингредиенты  
 3. Соблюдайте порядок закладки:  
 - Сначала твёрдые овощи  
 - Затем мягкие  
 - В конце зелень  
 4. Варите на медленном огне  
 5. Проверяйте готовность овощей  
 """  
  
 else -> "Неизвестное блюдо. Пожалуйста, выберите из списка: мясо, суп"  
 }  
}  
fun main() {  
 *println*("Введите тип блюда (мясо, суп):")  
 val food = *readLine*()  
 *println*(*getCookingInstructions*(food ?: ""))  
}



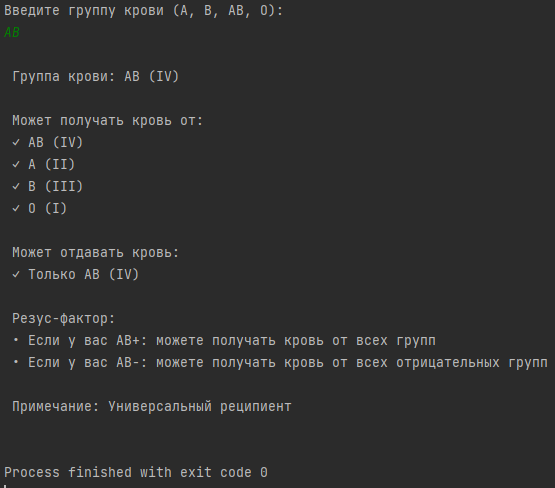
9. Способы оплаты: Напишите программу, которая принимает способ оплаты (наличные, кредитная карта, PayPal) и выдает соответствующее сообщение:

fun processPayment(paymentMethod: String): String {  
 return when(paymentMethod.*lowercase*()) {  
 "наличные" -> """  
 Оплата наличными:  
 1. Сумма к оплате будет указана на кассе  
 2. Подготовьте точную сумму или близкую к ней  
 3. Получите сдачу и кассовый чек  
 4. Проверьте правильность суммы и сдачи  
 Комиссия: нет  
 Время обработки: моментально  
 """  
 "кредитная карта", "карта" -> """  
 Оплата картой:  
 1. Вставьте карту в терминал или приложите к считывателю  
 2. Выберите способ оплаты (кредит/дебет)  
 3. Введите PIN-код при необходимости  
 4. Дождитесь подтверждения операции  
 5. Сохраните чек  
 Комиссия: согласно условиям банка  
 Время обработки: 1-2 минуты  
 """  
 "paypal" -> """  
 Оплата через PayPal:  
 1. Войдите в свой аккаунт PayPal  
 2. Проверьте email получателя платежа  
 3. Укажите сумму перевода  
 4. Выберите источник средств  
 5. Подтвердите транзакцию  
 Комиссия: 0-2% в зависимости от типа операции  
 Время обработки: 1-24 часа  
 """  
  
 else -> """  
 Доступные способы оплаты:  
 1. Наличные  
 2. Кредитная карта  
 3. PayPal  
  
 Пожалуйста, выберите один из указанных способов оплаты.  
 """  
 }  
}  
fun main() {  
 *println*("Выберите способ оплаты (наличные, кредитная карта, PayPal):")  
 val paymentMethod = *readLine*()  
 *println*(*processPayment*(paymentMethod ?: ""))  
}



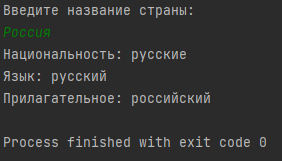
10. Группа крови: вводится группа крови (A, B, AB, O) и выводится, какие типы крови можно применять для переливания:

fun getBloodTypeCompatibility(bloodType: String): String {  
 return when(bloodType.*uppercase*()) {  
 "A", "А" -> """  
 Группа крови: A (II)  
  
 Может получать кровь от:  
 ✓ A (II)  
 ✓ O (I)  
  
 Может отдавать кровь:  
 ✓ A (II)  
 ✓ AB (IV)  
  
 Резус-фактор:  
 • Если у вас A+: можете получать A+, A-, O+, O-  
 • Если у вас A-: можете получать только A-, O-  
 """  
 "B", "В" -> """  
 Группа крови: B (III)  
  
 Может получать кровь от:  
 ✓ B (III)  
 ✓ O (I)  
  
 Может отдавать кровь:  
 ✓ B (III)  
 ✓ AB (IV)  
  
 Резус-фактор:  
 • Если у вас B+: можете получать B+, B-, O+, O-  
 • Если у вас B-: можете получать только B-, O-  
 """  
 "AB", "АВ" -> """  
 Группа крови: AB (IV)  
  
 Может получать кровь от:  
 ✓ AB (IV)  
 ✓ A (II)  
 ✓ B (III)  
 ✓ O (I)  
  
 Может отдавать кровь:  
 ✓ Только AB (IV)  
  
 Резус-фактор:  
 • Если у вас AB+: можете получать кровь от всех групп  
 • Если у вас AB-: можете получать кровь от всех отрицательных групп  
  
 Примечание: Универсальный реципиент  
 """  
 "O", "О", "0" -> """  
 Группа крови: O (I)  
  
 Может получать кровь от:  
 ✓ Только O (I)  
  
 Может отдавать кровь:  
 ✓ O (I)  
 ✓ A (II)  
 ✓ B (III)  
 ✓ AB (IV)  
  
 Резус-фактор:  
 • Если у вас O+: можете получать O+, O-  
 • Если у вас O-: можете получать только O-  
  
 Примечание: Универсальный донор  
 """  
 else -> """  
 Ошибка ввода!  
  
 Пожалуйста, введите корректную группу крови:  
 • A (или А)  
 • B (или В)  
 • AB (или АВ)  
 • O (или О, 0)  
  
 Важно: При переливании крови также учитывается резус-фактор!  
 """  
 }  
}  
fun main() {  
 *println*("Введите группу крови (A, B, AB, O):")  
 val bloodType = *readLine*()  
 *println*(*getBloodTypeCompatibility*(bloodType ?: ""))  
}



11. Национальности: Напишите программу, которая по странам (США, Россия, Япони и тд.):

fun getNationalityInfo(country: String): String {  
 return when(country.*lowercase*()) {  
 "россия" -> "Национальность: русские\nЯзык: русский\nПрилагательное: российский"  
 "сша", "соединенные штаты америки" -> "Национальность: американцы\nЯзык: английский\nПрилагательное: американский"  
 "япония" -> "Национальность: японцы\nЯзык: японский\nПрилагательное: японский"  
 "китай" -> "Национальность: китайцы\nЯзык: китайский\nПрилагательное: китайский"  
  
 else -> "Страна не найдена. Пожалуйста, проверьте правильность написания."  
 }  
}  
fun main() {  
 println("Введите название страны:")  
 val country = readLine()  
 *println*(*getNationalityInfo*(country ?: ""))  
}



12. Коды ошибок: Программа принимает код ошибки (100, 200, 300) и выводит сообщение об ошибке (например, "Ошибка сети", "Ошибка сервера" и т.д.):

fun getErrorDescription(errorCode: Int): String { return when(errorCode) { // 1xx - Информационные

100 -> "100 Continue: Продолжение запроса" 101 -> "101 Switching Protocols: Переключение протоколов"

102 -> "102 Processing: Идёт обработка" 103 -> "103 Early Hints: Ранние подсказки" // 2xx - Успешные

200 -> "200 OK: Успешный запрос"

201 -> "201 Created: Ресурс создан"

202 -> "202 Accepted: Запрос принят"

203 -> "203 Non-Authoritative Information: Информация не авторитетна"

204 -> "204 No Content: Нет содержимого"

205 -> "205 Reset Content: Сбросить содержимое"

206 -> "206 Partial Content: Частичное содержимое"

207 -> "207 Multi-Status: Многостатусный" // 3xx - Перенаправления

300 -> "300 Multiple Choices: Множественный выбор"

301 -> "301 Moved Permanently: Ресурс перемещён навсегда"

302 -> "302 Found: Ресурс временно перемещён"

303 -> "303 See Other: Смотреть другое"

304 -> "304 Not Modified: Не изменялось"

305 -> "305 Use Proxy: Использовать прокси"

307 -> "307 Temporary Redirect: Временное перенаправление"

308 -> "308 Permanent Redirect: Постоянное перенаправление" // 4xx - Ошибки клиента

400 -> "400 Bad Request: Некорректный запрос"

401 -> "401 Unauthorized: Не авторизован"

402 -> "402 Payment Required: Требуется оплата"

403 -> "403 Forbidden: Доступ запрещён"

404 -> "404 Not Found: Ресурс не найден"

405 -> "405 Method Not Allowed: Метод не поддерживается"

406 -> "406 Not Acceptable: Неприемлемый запрос"

407 -> "407 Proxy Authentication Required: Требуется аутентификация прокси"

408 -> "408 Request Timeout: Истекло время ожидания"

409 -> "409 Conflict: Конфликт"

410 -> "410 Gone: Ресурс удалён"

411 -> "411 Length Required: Необходима длина"

412 -> "412 Precondition Failed: Условие ложно"

413 -> "413 Payload Too Large: Полезная нагрузка слишком велика"

414 -> "414 URI Too Long: URI слишком длинный"

415 -> "415 Unsupported Media Type: Неподдерживаемый тип данных"

416 -> "416 Range Not Satisfiable: Диапазон не достижим"

417 -> "417 Expectation Failed: Ожидание не удалось"

418 -> "418 I'm a teapot: Я - чайник"

429 -> "429 Too Many Requests: Слишком много запросов" // 5xx - Ошибки сервера

500 -> "500 Internal Server Error: Внутренняя ошибка сервера"

501 -> "501 Not Implemented: Не реализовано"

502 -> "502 Bad Gateway: Плохой шлюз"

503 -> "503 Service Unavailable: Сервис недоступен"

504 -> "504 Gateway Timeout: Шлюз не отвечает"

505 -> "505 HTTP Version Not Supported: Версия HTTP не поддерживается"

506 -> "506 Variant Also Negotiates: Вариант также проводит согласование"

507 -> "507 Insufficient Storage: Недостаточно места"

508 -> "508 Loop Detected: Обнаружено бесконечное перенаправление"

510 -> "510 Not Extended: Не расширено"

511 -> "511 Network Authentication Required: Требуется сетевая аутентификация"

else -> """ Код ошибки не найден!

Основные группы кодов:

1xx - Информационные

2xx - Успешные

3xx - Перенаправления

4xx - Ошибки клиента

5xx - Ошибки сервера

Пожалуйста, введите корректный код ошибки.

"""

}

} fun main() {

println("Введите код ошибки: ")

val errorCode = readLine()?.toIntOrNull()

if (errorCode != null) {

println(getErrorDescription(errorCode))

} else {

println("Пожалуйста, введите числовой код ошибки")

}

}

