

Содержание

| | |
|--------------------------------------|-----------|
| Предисловие | 11 |
| Происхождение Go | 12 |
| Проект Go | 13 |
| Структура книги | 15 |
| Дополнительная информация | 17 |
| Благодарности | 18 |
| Ждем ваших отзывов! | 19 |
| Глава 1. Учебник | 21 |
| 1.1. Hello, World | 21 |
| 1.2. Аргументы командной строки | 24 |
| 1.3. Поиск повторяющихся строк | 29 |
| 1.4. Анимированные GIF-изображения | 34 |
| 1.5. Выборка URL | 37 |
| 1.6. Параллельная выборка URL | 39 |
| 1.7. Веб-сервер | 41 |
| 1.8. Некоторые мелочи | 46 |
| Глава 2. Структура программы | 49 |
| 2.1. Имена | 49 |
| 2.2. Объявления | 50 |
| 2.3. Переменные | 52 |
| 2.3.1. Краткое объявление переменной | 53 |
| 2.3.2. Указатели | 54 |
| 2.3.3. Функция new | 57 |
| 2.3.4. Время жизни переменных | 58 |
| 2.4. Присваивания | 59 |
| 2.4.1. Присваивание кортежу | 60 |
| 2.4.2. Присваиваемость | 61 |
| 2.5. Объявления типов | 62 |
| 2.6. Пакеты и файлы | 64 |
| 2.6.1. Импорт | 66 |
| 2.6.2. Инициализация пакетов | 68 |
| 2.7. Область видимости | 70 |

| | |
|---|-----|
| Глава 3. Фундаментальные типы данных | 75 |
| 3.1. Целые числа | 75 |
| 3.2. Числа с плавающей точкой | 81 |
| 3.3. Комплексные числа | 86 |
| 3.4. Булевы значения | 88 |
| 3.5. Строки | 90 |
| 3.5.1. Строковые литералы | 91 |
| 3.5.2. Unicode | 92 |
| 3.5.3. UTF-8 | 93 |
| 3.5.4. Строки и байтовые срезы | 97 |
| 3.5.5. Преобразования между строками и числами | 101 |
| 3.6. Константы | 102 |
| 3.6.1. Генератор констант <code>iota</code> | 103 |
| 3.6.2. Нетипизированные константы | 105 |
| Глава 4. Составные типы | 109 |
| 4.1. Массивы | 109 |
| 4.2. Срезы | 112 |
| 4.2.1. Функция <code>append</code> | 117 |
| 4.2.2. Работа со срезами “на месте” | 120 |
| 4.3. Отображения | 123 |
| 4.4. Структуры | 130 |
| 4.4.1. Структурные литералы | 133 |
| 4.4.2. Сравнение структур | 134 |
| 4.4.3. Встраивание структур и анонимные поля | 135 |
| 4.5. JSON | 138 |
| 4.6. Текстовые и HTML-шаблоны | 144 |
| Глава 5. Функции | 151 |
| 5.1. Объявления функций | 151 |
| 5.2. Рекурсия | 153 |
| 5.3. Множественные возвращаемые значения | 157 |
| 5.4. Ошибки | 160 |
| 5.4.1. Стратегии обработки ошибок | 161 |
| 5.4.2. Конец файла (EOF) | 164 |
| 5.5. Значения-функции | 165 |
| 5.6. Анонимные функции | 168 |
| 5.6.1. Предупреждение о захвате переменных итераций | 174 |
| 5.7. Вариативные функции | 176 |
| 5.8. Отложенные вызовы функций | 178 |
| 5.9. Аварийная ситуация | 183 |
| 5.10. Восстановление | 186 |

| | |
|---|-----|
| Глава 6. Методы | 191 |
| 6.1. Объявления методов | 191 |
| 6.2. Методы с указателем в роли получателя | 194 |
| 6.2.1. Значение <code>nil</code> является корректным получателем | 196 |
| 6.3. Создание типов путем встраивания структур | 197 |
| 6.4. Значения-методы и выражения-методы | 200 |
| 6.5. Пример: тип битового вектора | 202 |
| 6.6. Инкапсуляция | 205 |
| Глава 7. Интерфейсы | 209 |
| 7.1. Интерфейсы как контракты | 209 |
| 7.2. Типы интерфейсов | 212 |
| 7.3. Соответствие интерфейсу | 213 |
| 7.4. Анализ флагов с помощью <code>flag.Value</code> | 217 |
| 7.5. Значения интерфейсов | 220 |
| 7.5.1. Осторожно: интерфейс, содержащий нулевой указатель, не является нулевым | 224 |
| 7.6. Сортировка с помощью <code>sort.Interface</code> | 225 |
| 7.7. Интерфейс <code>http.Handler</code> | 231 |
| 7.8. Интерфейс <code>error</code> | 236 |
| 7.9. Пример: вычислитель выражения | 238 |
| 7.10. Декларации типов | 246 |
| 7.11. Распознавание ошибок с помощью деклараций типов | 248 |
| 7.12. Запрос поведения с помощью деклараций типов | 250 |
| 7.13. Выбор типа | 252 |
| 7.14. Пример: XML-декодирование на основе лексем | 255 |
| 7.15. Несколько советов | 258 |
| Глава 8. Go-подпрограммы и каналы | 259 |
| 8.1. Go-подпрограммы | 259 |
| 8.2. Пример: параллельный сервер часов | 261 |
| 8.3. Пример: параллельный эхо-сервер | 265 |
| 8.4. Каналы | 267 |
| 8.4.1. Небуферизованные каналы | 269 |
| 8.4.2. Конвейеры | 270 |
| 8.4.3. Однонаправленные каналы | 273 |
| 8.4.4. Буферизованные каналы | 275 |
| 8.5. Параллельные циклы | 278 |
| 8.6. Пример: параллельный веб-сканер | 283 |
| 8.7. Мультиплексирование с помощью <code>select</code> | 288 |
| 8.8. Пример: параллельный обход каталога | 292 |
| 8.9. Отмена | 296 |
| 8.10. Пример: чат-сервер | 299 |

| | |
|--|-----|
| Глава 9. Параллельность и совместно используемые переменные | 303 |
| 9.1. Состояния гонки | 303 |
| 9.2. Взаимные исключения: <code>sync.Mutex</code> | 309 |
| 9.3. Мьютексы чтения/записи: <code>sync.RWMutex</code> | 313 |
| 9.4. Синхронизация памяти | 314 |
| 9.5. Отложенная инициализация: <code>sync.Once</code> | 316 |
| 9.6. Детектор гонки | 319 |
| 9.7. Пример: параллельный неблокирующий кеш | 319 |
| 9.8. Go-подпрограммы и потоки | 328 |
| 9.8.1. Растущие стеки | 328 |
| 9.8.2. Планирование go-подпрограмм | 329 |
| 9.8.3. GOMAXPROCS | 329 |
| 9.8.4. Go-подпрограммы не имеют идентификации | 330 |
| Глава 10. Пакеты и инструменты Go | 333 |
| 10.1. Введение | 333 |
| 10.2. Пути импорта | 334 |
| 10.3. Объявление пакета | 335 |
| 10.4. Объявления импорта | 336 |
| 10.5. Пустой импорт | 337 |
| 10.6. Пакеты и именование | 339 |
| 10.7. Инструментарий Go | 341 |
| 10.7.1. Организация рабочего пространства | 342 |
| 10.7.2. Загрузка пакетов | 343 |
| 10.7.3. Построение пакетов | 344 |
| 10.7.4. Документирование пакетов | 347 |
| 10.7.5. Внутренние пакеты | 349 |
| 10.7.6. Запрашиваемые пакеты | 350 |
| Глава 11. Тестирование | 353 |
| 11.1. Инструмент <code>go test</code> | 354 |
| 11.2. Тестовые функции | 354 |
| 11.2.1. Рандомизированное тестирование | 359 |
| 11.2.2. Тестирование команд | 361 |
| 11.2.3. Тестирование белого ящика | 363 |
| 11.2.4. Внешние тестовые пакеты | 367 |
| 11.2.5. Написание эффективных тестов | 369 |
| 11.2.6. Избегайте хрупких тестов | 371 |
| 11.3. Охват | 372 |
| 11.4. Функции производительности | 375 |
| 11.5. Профилирование | 378 |
| 11.6. Функции-примеры | 381 |

| | |
|---|-----|
| Глава 12. Рефлексия | 383 |
| 12.1. Почему рефлексия? | 383 |
| 12.2. <code>reflect.Type</code> и <code>reflect.Value</code> | 384 |
| 12.3. Рекурсивный вывод значения | 387 |
| 12.4. Пример: кодирование S-выражений | 393 |
| 12.5. Установка переменных с помощью <code>reflect.Value</code> | 396 |
| 12.6. Пример: декодирование S-выражений | 399 |
| 12.7. Доступ к дескрипторам полей структур | 403 |
| 12.8. Вывод методов типа | 407 |
| 12.9. Предостережение | 408 |
| Глава 13. Низкоуровневое программирование | 409 |
| 13.1. <code>unsafe.Sizeof</code> , <code>Alignof</code> и <code>Offsetof</code> | 410 |
| 13.2. <code>unsafe.Pointer</code> | 412 |
| 13.3. Пример: глубокое равенство | 415 |
| 13.4. Вызов кода C с помощью <code>cgo</code> | 418 |
| 13.5. Еще одно предостережение | 423 |
| Предметный указатель..... | 425 |