



04. Программирование на Ruby

Практические задания.

№ 04.1

1. В главном меню выберите раздел «Программирование» и запустите из него редактор **Geany**.
2. Создайте в редакторе **Geany** новый файл **cycles.rb** и введите в него с клавиатуры текст программы (или скопируйте из этого файла):

```
#!/usr/bin/ruby

minimum = 100          # сначала будет такое минимальное число
maximum = -1           # сначала будет такое максимальное число

25.times do |i|        # повторить 25 раз, помещая число в i
  r = rand(100)         # поместить в r случайное число от 0 до 99
  print i, ' ', r, "\n" # напечатать i и r
  if r < minimum        # сравнить, меньше ли r значения в minimum
    minimum = r         # если да, то поместить r в minimum
  end
  # из 2-х чисел [в, списке] выбрать максимум и поместить в maximum
  maximum = [maximum, r].max
  # minimum = [minimum, r].min # минимум можно вычислять так же
end

print "Минимальное случайное число = ", minimum, "\n"
print "Максимальное случайное число = ", maximum, "\n"
```

Эта программа с помощью метода `rand(100)` в цикле генерирует 25 случайных целых чисел от 0 до 99. При этом она сравнивает полученное случайное число с ранее сохранёнными в переменных `minimum` и `maximum` минимальным и максимальным значениями. Если очередное полученное случайное число меньше ранее найденного минимального, то оно присваивается как новое значение переменной `minimum`. Если очередное полученное случайное число больше ранее найденного максимального, то оно присваивается как новое значение переменной `maximum`. В конце программы найденные минимальные и максимальные значения случайных величин выводятся на экран.

3. Сохраните программу **cycles.rb** в рабочем каталоге `~/projects/`.
4. Запустите программу на выполнение из раздела меню «Сборка», пункт «Execute» (выполнить).
5. Понаблюдайте результаты её работы в открывшемся терминальном окне, запустив несколько раз. Не забывайте каждый раз закрывать терминальное окно.

№ 04.2

1. Напишите на **Ruby** программу **happy.rb**, которая в цикле находит все «счастливые» билеты в рулоне с номерами билетов от 000001 до 999999. Билет считается счастливым, если сумма первых трёх цифр равна сумме последних трёх цифр.
2. Сохраните свою программу в каталоге `~/projects/`.
3. Выводите все номера счастливых билетов на экран командой `print`.

- Чтобы запускать программу в терминальном окне, сделайте её исполняемой, установив признак «x» для всех (a) пользователей. Для этого выполните в терминальном окне команды:

```
cd ~/projects
chmod a+x happy.rb
```
- Чтобы направить этот вывод с экрана в файл, запустите эту программу из терминального окна с перенаправлением стандартного вывода с экрана в файл:

```
cd ~/projects
./happy.rb > happy.txt
```
- Подсчитайте общее количество найденных счастливых билетов и выведите его на экран.
- Можно также сосчитать количество счастливых билетов в каждой тысяче и вывести на экран их количество и процент на каждую тысячу.

№ 04.3

- Программа `~/CodeClub-IoT/samples/thermo.rb` на **Ruby** опрашивает 12 раз встроенный датчик температуры Raspberry Pi и выводит текущую дату, время и показания датчика.

```
require "thermal_sensor"

sensor = RaspberryPi::ThermalSensor.new # создать объект «датчик»

(1..12).each do |n| # повторить 12 раз
  sensor.read_data # прочитать показания датчика
  printf("Дата: %s Время: %s Температура: %7.4f°C\n", # строка вывода
    Time.now.strftime("%d.%m.%Y"), # текущая дата вставляется в строку
    Time.now.strftime("%H:%M:%S"), # текущее время вставляется в строку
    sensor.celsius) # значение температуры вставляется в строку
  sleep 1 # подождать 1 секунду
end
```

- Выполните программу из редактора **Geany** и наблюдайте результаты её работы в терминальном окне.
- Сохраните программу в рабочем каталоге `~/projects/` под именем `thermo_csv.rb`.
- Измените программу так, чтобы она выводила только одну запись с датой (ГГГГ-ММ-ДД), временем (ЧЧ:ММ:СС) и показанием температуры (ТТ.ТТТТ) в формате CSV:
ГГГГ-ММ-ДД, ЧЧ:ММ:СС, ТТ.ТТТТ
- Сделайте программу исполняемой (см. предыдущее задание), запустите её в терминальном окне и проверьте, как она работает:

```
cd ~/projects
./thermo_csv.rb
```
- Для запуска программы `thermo_csv.rb` напишите исполняемый командный скрипт, для чего скопируйте текст скрипта

```
#!/bin/bash
cd ~/projects
./thermo_csv.rb >> thermo_log.txt
```

в новый файл `thermo_csv.sh` в каталоге `~/projects/`.
- Добавьте скрипт `thermo_csv.sh` в расписание демона **cron** для автоматического запуска 1 раз в 5 минут, для чего запустите редактор расписания:

```
crontab -e
```

и введите нужное правило:

```
*/5 * * * * /home/pi/projects/thermo_csv.sh
```
- Проверьте, что всё работает правильно: строки с данными о температуре в требуемом формате добавляются в файл протокола `~/projects/thermo_log.txt` каждые 5 минут.

Подробнее о демоне **cron** и авто-запуске команд по расписанию — в файле `~/CodeClub-IoT/github/theory/IoT-Shell_scripts.pdf`.