Отчет по лабораторной работе 1

ИМПЕРАТИВНОЕ (ПРОЦЕДУРНОЕ) ПРОГРАММИРОВАНИЕ

Дата: 2025-09-10

Семестр: 2 курс 1 полугодие - 3 семестр

Группа: ПИН-б-о-24-1(1)

Дисциплина: Технологии программирования

Студент: Герда Никита Андреевич

Цель работы

Познакомиться с особенностями процедурного программирования. Решить задания в процедурном стиле. Составить отчет.

Теоретическая часть

Процедурный стиль предоставляет возможность программисту определять каждый шаг в процессе решения задачи. Переменная состоит из имени и выделенной области памяти, которая соответствует ей. Для объявления или, другими словами, создания переменной используются **директивы** (ключевые слова, конструкции).

Функция – это подпрограмма специального вида, которая может принимать на вход параметры, выполнять различные действий и передавать результаты работы.

Процедура – это независимая именованная часть программы, которую после однократного описания можно многократно вызвать по имени из последующих частей программы для выполнения определенных действий.

Практическая часть

Выполненные задачи

- [x] Задача 1: Написать программу, выполненную в процедурном стиле. Программа должна быть выполнена в виде псевдокода, в виде блок-схемы и на языке высокого уровня (ЯВУ)

(здесь и далее, если не оговорено иное, при отсылке к ЯВУ необходимо выполнять код на языке R). Для построения блоксхемы рекомендуется использовать ресурс draw.io или аналогичную программу. Построение блок схемы делается с учетом правил, содержащихся в презентации Императивное (процедурное) программирование. **Вариант 3** Напишите программу, подсчитывающую среднее количество занятий в неделю. Программа запрашивает информацию о количестве занятий в день (по дням недели). На выходе программа указывает среднее количество занятий в неделю с округлением до ближайшего целого числа.

- [x] Задача 2: Опишите, представленный код в виде псевдокода и ответьте на вопрос, что будет получено при передаче функции числа 7? Также реализуйте данный алгоритм на ЯВУ.

```
foo:
 cmp $0, %edi
 jg calc
 mov $1, %eax
 jmp exit
calc:
 push %edi
 sub $1, %edi
 call foo
 pop %edi
 imul %edi, %eax
exit:
 ret
```

Это функция с одним входным параметром, для которой ABI (двоичный интерфейс приложений) предписывает передачу одного параметра через регистр %edi, а передачу возвращаемого значения через регистр %eax. Замечания:

1. Команда 'imul src, dest' умножает src на dest и кладет результат в dest.
2. Не нужно думать о переполнении. Его здесь не будет.

Ключевые фрагменты кода

```
Начало
  sum = 0
  ДЛЯ i от 1 ДО 7 С ШАГОМ 1
      ввод lessons
      sum = sum + lessons
  avg = sum / 7
  округлить avg
  вывод avg
Конец
```

```
sum <- 0
for (i in 1:7) {
  lessons <- as.numeric(readline(paste("Введите количество занятий в день", i, ": ")))
  sum <- sum + lessons
```

```
}
avg <- sum / 7
rounded_avg <- round(avg)
cat("Среднее количество занятий в неделю:", rounded_avg, "\n")
```

Задание 2.

```
ФУНКЦИЯ foo(n)
    ЕСЛИ n > 0
        результат = n * foo(n - 1)
        ВЕРНУТЬ результат
    ИНАЧЕ
        ВЕРНУТЬ 1
КОНЕЦ ФУНКЦИИ
```

Результаты выполнения

Пример работы программы

```
Введите количество занятий в день 1: 3
Введите количество занятий в день 2: 4
Введите количество занятий в день 3: 2
Введите количество занятий в день 4: 5
Введите количество занятий в день 5: 3
Введите количество занятий в день 6: 0
Введите количество занятий в день 7: 1
Среднее количество занятий в неделю: 3
```

Тестирование

- [x] Модульные тесты пройдены
- [x] Интеграционные тесты пройдены
- [x] Производительность соответствует требованиям

Выводы

1. Процедурное программирование основано на последовательном выполнении операторов, которые преобразуют исходные данные в результаты через изменение состояния памяти.
2. Ключевые элементы процедурного стиля - переменные (хранят данные в памяти) и подпрограммы (функции и процедуры), которые позволяют структурировать код и избегать повторений.
3. Эволюция подходов шла от простых линейных программ с переходами (goto) к структурированному программированию с процедурами, что улучшило читаемость и поддерживаемость кода.

Ответы на контрольные вопросы

1. Хронология процедурных языков - **1950-е:** Fortran (1957), ALGOL (1958) - первые процедурные языки **1960-е:** COBOL (1959), BASIC (1964), PL/I (1964) **1970-е:** Pascal (1970), C (1972), Ada (1979) **1980-е:** Modula-2 (1978), C++ (1985) - добавляет ООП к процедурной основе Эволюция шла от машинно-ориентированных языков к более абстрактным, с улучшенными возможностями структурирования кода.
2. Спагетти-код (особенность и причины) - Код с хаотичной структурой, где поток управления сложно проследить из-за обилия переходов (goto). Напоминает запутанную тарелку спагетти. **Причины:**

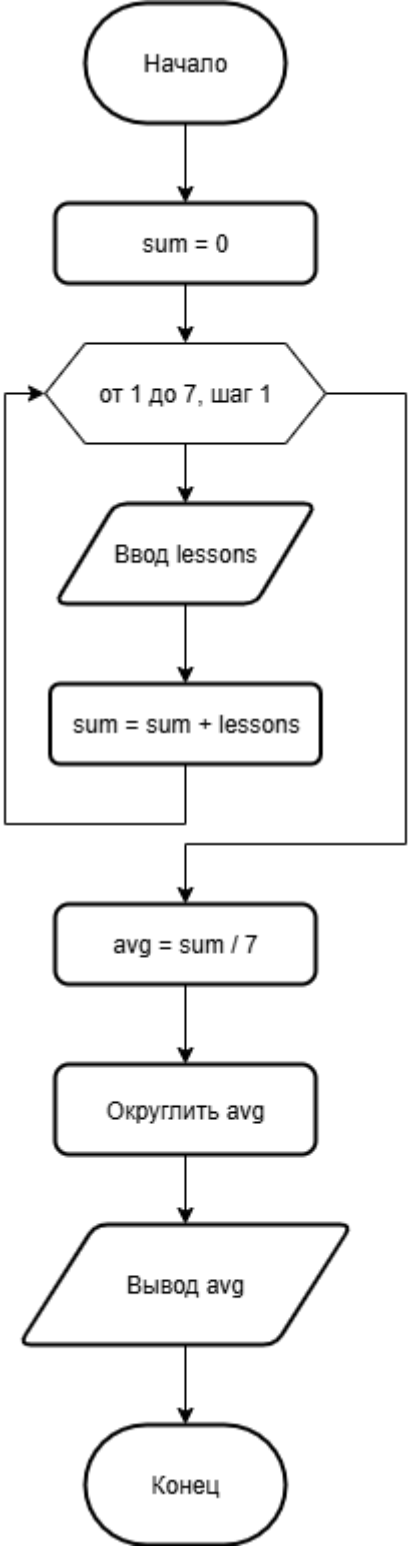
- Чрезмерное использование операторов безусловного перехода (goto) - Отсутствие структурированных конструкций (ветвлений, циклов)
- Непродуманная архитектура программы
- Многократные переходы между различными частями кода. Программы на ранних версиях BASIC с частыми GOTO создавали именно такой "спагетти-код".

3. Процедурный стиль и архитектура фон Неймана (взаимосвязь) - Процедурное программирование напрямую отражает архитектуру фон Неймана:

- Память - переменные в программе соответствуют ячейкам памяти
- Процессор - операторы программы соответствуют командам процессора
- Последовательное выполнение - отражает последовательную природу фон-неймановских вычислений
- Изменяемое состояние - соответствует изменению содержимого памяти

Приложения

- Исходный код
-


Начало
sum = 0
от 1 до 7, шаг 1
Ввод lessons
sum = sum + lessons
avg = sum / 7
Округлить avg
Вывод avg
Конец