ЛАБОРАТОРНАЯ РАБОТА №1
ИМПЕРАТИВНОЕ (ПРОЦЕДУРНОЕ) ПРОГРАММИРОВАНИЕ

Цель: познакомиться с особенностями процедурного программирования. Решить задания в процедурном стиле. Составить отчет.

Теоретические сведения

Выполнение программы в процедурном стиле сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти, т.е. значений исходных данных, в заключительное, т.е. в результаты. Таким образом, с точки зрения программиста имеются программа и память, причем первая последовательно обновляет содержимое последней.

Процедурный стиль предоставляет возможность программисту определять каждый шаг в процессе решения задачи. Особенность таких языков программирования состоит в том, что задачи разбиваются на шаги и решаются шаг за шагом. Используя процедурный язык, программист определяет языковые конструкции для выполнения последовательности алгоритмических шагов.

Переменная состоит из имени и выделенной области памяти, которая соответствует ей. Для объявления или, другими словами, создания переменной используются директивы (ключевые слова, конструкции). В разных языках программирования создание переменных отличается.

Функция – это подпрограмма специального вида, которая может принимать на вход параметры, выполнять различные действий и передавать результаты работы. Вызов функции является, с точки зрения языка программирования, выражением, он может использоваться в других выражениях или в качестве правой части присваивания.

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

```r
chr_var <- "Привет, Мир!" #Создание переменной строкового типа (character)
num_var <- 24.3 #Создание переменной вещественного типа (numeric)
int_var <- 124L #Создание переменной целочисленного типа (integer)
cplx_var <- 56 + 4i #Создание переменной комплексного типа (complex)
logi_var <- TRUE #Создание переменной логического типа (logical)
assign("name_var", "Какое-то значение") #Создание переменной с помощью assign()
```

Рисунок 1 – Создание переменных в языке R

Функции. Для создания функций используются служебные слова: func, def, function и др. Синтаксис создания функций отличается в разных языках программирования.

```r
#Создание функции, возвращающей большее значение
bigger <- function(a, b) {
  if (a > b) {
    return(a)
  } else {
    return(b)
  }
}
bigger(5, 8) #Корректное использование функции
bigger(5) #Не корректное использование функции
bigger(7, 6, 9) #Не корректное использование функции
```

Рисунок 2 – Создание и использование функций в языке R

Написание программ

С развитием языков программирования развивались и технологии, используемые при написании программного кода. Первые программы писались сплошным текстом. Это была простая последовательность команд. Все это выглядело следующим образом:

```
Начало программы
Команда 1
Команда 2
...
Команда N
Конец программы
```

Рисунок 3 – Структура линейной программы

Используя такой подход к программированию, можно было сделать очень мало. Единственное, что было доступно программисту для создания логики в данном случае – это переходы. Под переходом понимается переход на какую-то команду при определенных условиях, которые сложились в процессе обработки данных на процессоре.

```
Начало программы
Команда 1
Команда 2
Если выполнено условие, то вернуться к команде 1
Команда 3
...
Команда N
Конец программы
```

Рисунок 4 – Образец части программы, построенной с использованием безусловного перехода

Безусловный переход (goto, jmp).

Как правило, оператор безусловного перехода состоит из двух частей: собственно оператора и метки, указывающей целевую точку перехода в программе (например: goto метка). Метка, в зависимости от правил языка, может быть либо числом (как, например, в классическом BASIC), либо идентификатором используемого языка программирования. Для меток-идентификаторов метка, как правило, ставится перед оператором, на который должен осуществляться переход, и отделяется от него двоеточием (метка:).

Действие оператора перехода состоит в том, что после его исполнения следующими будут исполняться операторы программы, идущие в тексте непосредственно после метки (до следующего оператора перехода, ветвления или цикла). Для машинных языков инструкция перехода копирует в регистр процессора, содержащий адрес следующей выполняемой команды, адрес команды, помеченной меткой.

```
1 i=0
2 i+=1
3 PRINT i; "squared="; i*i
4 IF i>100 THEN GOTO 6
5 GOTO 2
6 PRINT "Program Completed."
7 END
```

а – процедурный стиль,

```
1 FOR i=1 TO 100
2     PRINT i; "squared="; i*i
3 NEXT i
4 PRINT "Program Completed."
5 END
```

б – структурное программирование

Рисунок 5 – Пример программ, печатающих числа от 1 до 100 и квадраты этих чисел, выполненные в разных стилях на языке программирования BASIC

В процедурном подходе какой-то код программы мог объединяться в отдельные блоки (процедуры). После этого такой блок команд можно вызывать из любой части программы.

```
Начало процедуры 1
Команда 1
Команда 2
Конец процедуры 1
Начало программы
Команда 1
Команда 2
Если выполнено условие, то выполнить код процедуры 1.
Команда 3
Конец программы.
```

Рисунок 6 – Структура программы, выполненной в процедурном стиле

Практическая часть

Задание 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 блок-схемы рекомендуется использовать ресурс draw.io или аналогичную программу. Построение блок схемы делается с учетом правил, содержащихся в презентации Императивное (процедурное) программирование.

Вариант 1

Напишите программу, рассчитывающую площадь трех фигур: квадрат, прямоугольник и круг. На входе программа запрашивает введение данных о фигурах (для квадрата – сторона, круг – радиус, прямоугольник – две стороны). На выходе программа указывает площади трех фигур и общую площадь.

Вариант 2

Напишите программу, рассчитывающую сумму расходов за месяц. На входе программа запрашивает сведения о расходах по нескольким пунктам (минимум 3 статьи расходов). На выходе программа указывает суммарные расходы и максимальную статью расходов.

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.

Задание 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.

Функция на ассемблере с синтаксисом AT&T:

```
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.

Контрольный вопросы:

1. Особенности процедурного программирования
2. Линейная программа
3. Понятия: переменная, процедура, функция,
4. Безусловный оператор

Вопросы для поиска и письменного ответа:

1. Хронология процедурных языков
2. Спагетти-код (особенность и причины)
3. Процедурный стиль и архитектура фон Неймана (взаимосвязь)