УДК 539
Damir Minibaev[1], Olha Kalantaieva[2]
[1]student of group CST-118 ZNTU
[2]teacher ZNTU

PROCEDURAL PROGRAMMING

First, a procedural language is a type of imperative language. That means that execution is based on statements. The alternative programming paradigm is declarative programming, in which execution is based on expressions. Early imperative languages included BASIC and COBOL. HTML and SQL are popular declarative languages.

A procedural program general includes a single 'main' instruction flow and a list of supporting functions that are called by the 'main', by other supporting functions and in some cases by themselves. These functions almost always return to the calling code; occasionally a function will exit the program, but this is the exception.

Structured languages have a set of global variables, and local variables for the supporting functions. Code and variables are isolated, not encapsulated.

The practical opposite of procedural languages is object-oriented languages. Executing a series of statements or expressions can be very limiting in itself. There needs to be a way to organize those statements/expressions. This led to the development of subroutines, which appear in both imperative and declarative languages.

In imperative languages, subroutines are called procedures.

In declarative languages, subroutines are called functions.

The modular programming movement encouraged programmers to package their code into independent modules that could be reused and reattached in unlimited ways. This was a precursor to object oriented programing, the extremely influential paradigm that began with SmallTalk and became most popular in the form of Java. In OOP, procedures are packaged along with the data they operate on

inside of objects. A procedure that belongs to an object is also referred to as a method.

Procedures form the basis of procedural programing, which is the style you asked about. Procedures are like mini programs that can be called from anywhere else in the overall program. After they complete, execution returns to the line after the procedure call (the return address). The return behavior is what distinguishes procedures from macros (an earlier concept of subroutine). C and Go are both procedural languages. The declarative counterpart to the procedure is the function, which forms the basis of functional programming. A pure function is like the mathematical concept of a function. You call it with arguments and it returns a value. There are no side effects, such as database reads, file system writes, or global state changes. Just a simple return value that is entirely based on its input. Haskell and Scala are both functional languages.

Historically, procedural programming arose out of the concept of structured programing. The idea is that programs can be written with only three things:

1. Sequence (execution moves forward one statement at a time);
2. Selection (if statements);
3. Iteration (loops).