Goto statement

GoTo ment A goto ment induces your program to unreservedly direct the control to the ment integrated with the label assigned to the goto statement. It is considered to be a basic operation for passing the control from one segment of your program to another one.

Although a goto statement represents a simple way of traveling around a program, it is generally regarded that excessive using goto statements may be fraught with serious consequences in a program as it usually results in problems while reading code, generally known as a spaghetti code, making it not only unreadable but also unmaintainable (Tribble1).

A goto statement is a constituent of many languages like C, C++, COBOL, Pascal, Algol and others, especially of assembly languages. Nevertheless a goto statement is not popular in all higher-level languages of programming. Thus in Java goto statement is a reserved statement at present (Tribble 3). During the period of early stage of structured programming development many experts in computer science arrived at a conclusion that in programs it is better to use " structured flow-control commands", for example, loops and " if-then-else statements" rather than a goto statement (Tribble 5). Though there are experts who consider that despite the fact that " goto statement considered harmful", there exist some problems in a great number of programming languages that can't be directly decided without a goto statement, like, for instance, exception handling or breaking out of nested loops (Knuth 268).

" There are few good uses for a goto statement. It is not uncommon for the class instructor to ban goto statements altogether. But the traditional legitimate use for a goto is to allow the programmer to escape from deep nesting when a special case (usually an error) has been encountered. For

Pascal programmers, this means that, on rare occasion, you may want to goto'' the end of a subroutine in order to exit. For C programmers, such a goto is never justified since you can return from any point in a subroutine, and the keywords break and continue allow you to break out of a loop. FORTRAN programs may use goto's to simulate while statements, or C-style break and continue statements" (Tribble 4).

The question of goto-less programming became of a great concern in 1968 when a letter by Edsger Dijkstra appeared. " For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of goto statements in the programs they produce. More recently I discovered why the use of the goto statement has such disastrous effects, and I became convinced that the goto statement should be abolished from all " higher level" programming languages (i. e. everything except, perhaps, plain machine code)" (Dijkstra 147).

Dijkstra and his followers consider that it is necessary to revoke unlimited using of goto statement from high-level languages as it results in problems while understanding and proving programs as well as in the process of debugging or modifying. Among disadvantages of using goto statement in your program is also the high possibility of creating an infinite loop. Dijkstra also maintains that goto statements may only be reasonable for " alarm exits", which now we name " fatal exceptions" (Tribble 7). Thus he agrees that in those languages which doesn't have a strong mechanism of exception handling, goto statements may become the only reasonable practical substitute. Besides Dijkstra refers to the proposition made by Hoare and Wirth about the " construction of the case/select control flow structure" (417). Nowadays its usefulness is proved, but at that time it was a question

of a great concern. Dijkstra emphasizes on its possibility to be used as the best alternative to a great number of goto statements (Tribble 5). " He is in fact arguing that some gotos in a program may be useful and may actually make the program easier to understand. Soit is safe to say that Dijkstra considered goto statements to be harmful, but not lethal, and certainly not useless" (Tribble 6). Thus the best decision of the controversy would be to know " when to use goto statement for good and when not to use it for evil" (Tribble 7).

Works cited

Dijkstra, Edsger. " GOTO Statement Considered Harmful." Letter of the Editor. Communications of the ACM, March 1968: 147-148.

Knuth, Donald. " Structured Programming with Goto Statements". Computing Surveys 4 (1974): 261-301.

Tribble, David. Go To Statement Considered Harmful: A Retrospective. Revision 1. 1, 27 November 2005. 21 February 2006 http://david. tribble. com/text/goto. html

Wirth, Niklaus and Hoare C. A.. " A contribution to the development of ALGOL". Communications of the ACM, 9 June 1966: 413-432.