5.4 — Goto statements

BY ALEX ON JUNE 21ST, 2007 | LAST MODIFIED BY ALEX ON JANUARY 23RD, 2020

The **goto statement** is a control flow statement that causes the CPU to jump to another spot in the code. This spot is identified through use of a **statement label**. The following is an example of a goto statement and statement label:

```
#include <iostream>
#include <cmath> // for sqrt() function

int main()
{
    double x;
tryAgain: // this is a statement label
    std::cout << "Enter a non-negative number";
    std::cin >> x;

    if (x < 0.0)
        goto tryAgain; // this is the goto statement

    std::cout << "The sqrt of " << x << " is " << sqrt(x) << std::endl;
    return 0;
}
```

In this program, the user is asked to enter a non-negative number. However, if a negative number is entered, the program utilizes a goto statement to jump back to the tryAgain label. The user is then asked again to enter a new number. In this way, we can continually ask the user for input until he or she enters something valid.

In the section on variables, we covered three kinds of scope: local (block) scope, file scope, and global scope. Statement labels utilize a fourth kind of scope: function scope. The goto statement and its corresponding statement label must appear in the same function.

There are some restrictions on the use of goto statements. For example, you can’t jump forward over a variable that’s initialized in the same block as the goto:

```
int main()
{
    goto skip; // invalid forward jump
    int x = 5;
skip:
    x += 3; // what would this even evaluate to?
    return 0;
}
```

In general, use of goto is shunned in C++ (and most other high level languages as well). **Edsger W. Dijkstra**, a noted computer scientist, laid out the case in a famous but difficult to read paper called **Go To Statement Considered Harmful**. The primary problem with goto is that it allows a programmer to cause the point of execution to jump around the code arbitrarily. This creates what is not-so-affectionately known as spaghetti code. **Spaghetti code** is code that has a path of execution that resembles a bowl of spaghetti (all tangled and twisted), making it extremely difficult to follow the logic of such code.

As Dijkstra says somewhat humorously, “the quality of programmers is a decreasing function of the density of go to statements in the programs they produce”.

Goto statements are common in some older languages, such as Basic or Fortran, and even used in C. However, in C++, goto statements are almost never used, as almost any code written using a goto statement can be more clearly written using other constructs in C++, such as loops, exception handlers, or destructors (all of which we'll cover in future lessons).

Rule: Avoid use of goto statements unless necessary

5.5 -- While statements

Index

5.3 -- Switch statements

C++ TUTORIAL | PRINT THIS POST

63 comments to 5.4 — Goto statements

« Older Comments 1 2

SM.Haider
June 17, 2019 at 5:57 pm · Reply

I have a question or two;
Once you initialize a variable, (obviously) it allocates a specific quantity of memory, so how many bytes of memory does a statement label allocate?
Secondly, if we utilised numerous goto statements in a program, will that delay the performance of the processor?

Bronzdragon
June 18, 2019 at 4:51 am · Reply

Statement labels aren't actually variables, so they're not allocated the same way.

To keep it simple, when the compiler compiles your code, it converts your programme to byte code. Each statement translates to 1 or more instructions (or sometimes zero, due to optimization). Each instruction has a position in the code. When you create a statement label, the compiler remembers the offset to the next instruction. When you jump, it takes the address of the start of your programme, and the offset, and jumps there.

This offset is stored directly in the instruction, it's not in any way dynamic, like variable decelerations are normally.

SM.Haider
June 18, 2019 at 6:48 am · Reply

Thanks.

June
June 24, 2019 at 2:41 am · Reply

Hmm to put it another way, I think the best analogy is a bookmark. To keep things simple, when you compile your program it places a bookmark next to the code beneath it. This is all the label amounts to. It doesn't make your program any bigger or smaller and doesn't take up any memory. It's just a simple bookmark that the compiler uses so that it knows where exactly to make the program jump to. Theres a lot more but I'm keeping things very simple.

Jumping around is very crucial to some languages, notably one of my favorites, assembly. However that's not really the way of C++ outside very special cases you won't hardly encounter. If you feel like a goto statement would work well then it's usually the case to sit back a minute and rethink your program so far. Of course you can always play around with goto to delve a bit into curiosity just know that in normal C++ programs it's not really expected to be used.

I hope I answered your question well ^_^

SM.Haider
June 24, 2019 at 2:57 am · Reply

Thank-you.

« Older Comments 1 2