# Flow (for)

void main() {

// :We do this kind of thing over and over and over ...

int x=0; // Initialization

while(x<6) { // Condition

cout << x << endl; // Code

x=x+1; // Update

}

// :The language allows a shortcut for it with the for loop.

// Initialization ; condition; update

for( int x=0; x<6; x=x+1) {

cout << x << endl; // Code

}

// :The "condition" is checked at the top (first time through)

// :The "update" happens last

// : IMPORTANT. The "continue" executes the "update" on the way back

// to the expression. In the "while" above the "update" is NOT

// executed.

system("pause");

}

for(int x=0;x<5;) { // Any part can be left out

cout << x << endl;

x=x+1;

}

// :If expression is left out then it is "1"

for(;;) { } // Common infinite loop

for(int x=0;x<5;++x) { // Does ++x or x++ matter? (Think of substitution)

cout << x << endl;

}

// :Can do multiple steps in init and update. Only one expression.

// :Separate with coma

for(int x=0,doStuff(),y=4;x<0;++x,++y,doOther()) {

}

## Switch

int x;

cin >> x;

// : This is common. You want to select one of several blocks of code.

if(x==0) {

cout << "Zero" << endl;

} else if(x==1) {

cout << "One" << endl;

} else {

cout << "Something else" << endl;

}

// : With the if/else it takes a long time to get to the last block. The

// more you have the longer it takes.

// : Better to tell the compiler about this situation. Tell it which

// variable is the controller and what all the blocks of code are.

switch(x) { // X is the controller. The "work" happens here at this line.

case 0: // X==0

cout << "Zero" << endl;

break; // Required or the CPU will "fall through" to next line.

case 1: // X==1

cout << "One" << endl;

break;

default: // Everything not listed

cout << "Something else" << endl;

break;

}

// : If you have a lot of cases then the generated code will use a

// table of pointers such that the selection takes the same time

// for each option.

// The "case 1:" is NOT code. It is a label. The switch finds the right

// label and jumps to it. Thus you need the "break".

switch(x) {

case -20: // Can be negative

case -14: // Empty labels fall through to a line of code

case 5:

cout << "Here" << endl;

// No break here ... falls into case 0

case 0:

cout << "Zero" << endl;

break;

case 1000: // Do not have to be in order of complete

cout << "BIG" << endl;

break;

case 1:

cout << "One" << endl;

break;

default:

cout << "Something else" << endl;

}

The compiler generates the best code. Several “if/else” might be the best. Maybe different for different parts of the range.

Because it is jump-table based MUST BE INTEGER. No floats … no strings.

In the debugger you can show the disassembly and see what is happening. Can put breakpoints on the assembly and step.

switch(x) {

004113EE mov eax,dword ptr [x] ; Value of X to eax register

004113F1 mov dword ptr [ebp-0DCh],eax ; Hold in temporary

004113F7 cmp dword ptr [ebp-0DCh],3 ; Unsigned value beyond 3?

004113FE ja $LN2+7 (41142Dh) ; Jump-if-above to default

00411400 mov ecx,dword ptr [ebp-0DCh] ; Temporary to ecx register

00411406 jmp dword ptr (4114BCh)[ecx\*4] ; Jump table (4 byte entries)

case 0:

y = 10;

0041140D mov dword ptr [y],0Ah ; Set Y to 10

break;

00411414 jmp $LN2+0Eh (411434h) ; Jump beyond loop

case 1:

y = 11;

00411416 mov dword ptr [y],0Bh ; Set Y to 11

break;

0041141D jmp $LN2+0Eh (411434h) ; Jump beyond loop

case 2:

y=12;

0041141F mov dword ptr [y],0Ch

case 3:

y=13;

00411426 mov dword ptr [y],0Dh

default:

y=20;

0041142D mov dword ptr [y],14h

}

cout << y << endl;

00411434 mov esi,esp

004114BC db 0dh ; LSB first 0041140D

004114BD db 14h

004114BE db 41h

004114BF db 00h

004114C0 db 16h ; LSB first 00411416

004114C1 db 14h

004114C2 db 41h

004114C3 db 00h