

# Operatoren und Ausdrücke

Montag, 25. März 2019 19:51

```

int i;
int * qi;
int * zi;

qi = sv2.a;
002518D5 8D 85 4C FC FF FF lea    eax,[ebp-3B4h]
002518DB 89 85 34 FC FF FF mov    dword ptr [ebp-3CCh],eax
zi = sv1.a;
002518E1 8D 85 E4 FD FF FF lea    eax,[ebp-21Ch]
002518E7 89 85 28 FC FF FF mov    dword ptr [ebp-3D8h],eax
for (i = 100; i > 0; i--, qi++, zi++)
002518ED C7 85 40 FC FF FF 64 00 00 00 mov    dword ptr [ebp-3C0h],64h
002518F7 EB 2D          jmp    main+1B6h (0251926h)
002518F9 8B 85 40 FC FF FF mov    eax,dword ptr [ebp-3C0h]
002518FF 83 E8 01          sub    eax,1
for (i = 100; i > 0; i--, qi++, zi++)
00251902 89 85 40 FC FF FF mov    dword ptr [ebp-3C0h],eax
00251908 8B 8D 34 FC FF FF mov    ecx,dword ptr [ebp-3CCh]
0025190E 83 C1 04          add    ecx,4
00251911 89 8D 34 FC FF FF mov    dword ptr [ebp-3CCh],ecx
00251917 8B 95 28 FC FF FF mov    edx,dword ptr [ebp-3D8h]
0025191D 83 C2 04          add    edx,4
00251920 89 95 28 FC FF FF mov    dword ptr [ebp-3D8h],edx
00251926 83 BD 40 FC FF FF 00 cmp    dword ptr [ebp-3C0h],0
0025192D 7E 12          jle    main+1D1h (0251941h)
*zi = *qi;
0025192F 8B 85 28 FC FF FF mov    eax,dword ptr [ebp-3D8h]
00251935 8B 8D 34 FC FF FF mov    ecx,dword ptr [ebp-3CCh]
0025193B 8B 11          mov    edx,dword ptr [ecx]
0025193D 89 10          mov    dword ptr [eax],edx
0025193F EB B8          jmp    main+189h (02518F9h)
}
return 0;
00251941 33 C0          xor    eax,eax
    
```

*Handwritten annotations:*

- SV2.a* points to `qi = sv2.a;`
- SV1.a* points to `zi = sv1.a;`
- zi* points to `int * zi;`
- i* points to `int i;`
- i = 100* is written next to the loop initialization.
- i--* is written next to the first `jle` instruction.
- qi++* and *zi++* are written next to the `mov` instructions for `ecx` and `edx`.
- if (i <= 0) gehe zu Ende* is written next to the `jle` instruction.
- lese qi → ECX* and *zi → EAX* are written next to the `mov` instructions for `ecx` and `edx`.
- lies \*qi* and *Schreibe \*zi* are written next to the `mov` instructions for `edx` and `eax`.

*for (i=100; i > 0; i--)*

*Rumpf*

*i = 100;*

*Bedingung: if (i > 0) goto Endp* *100 + 1*

*Rumpf*

*i--;*

*goto Bedingung;* *100*

*Ende:*

*201*

*i = 100;*

*goto Bedingung; 101*

*Anfang:*

*Rumpf*

*i--;*

*Bedingung:*

*if i > 0 goto Anfang; 100 + 1*

*102*