Programmētāju skola
3. līmeņa grupa

Atkļūdošana

Atkļūdošana

Bug kļūda, kas programmētāja vai shēmtehniķa neuzmanības dēļ rodas programmas vai shēmas sastādīšanas procesā.

Debugging (*atkļūdošana*) procedūra programmas sastādīšanas vai shēmas izstrādāšanas gaitā pieļauto kļūdu atrašanai, lokalizēšanai un novēršanai.

Debugger (*atkļūdotājs*) programma, kas palīdz programmētājiem veikt atkļūdošanu. Atkļūdotājs parasti satur tādus rīkus kā: pārtraukumpunkti (breakpoints), uzraudzīšana (watch) un trasēšana (trace).

Debug build (*atkļūdošanas versija*) programmas versija ar speciālo informāciju, kas ir nepieciešama atkļūdošanas veikšanai. Programmētāji izmanto tādu versiju programmas izstrādāšanas laikā.

Release build (*izlaišanas versija*) izpildīšanas ātruma un koda apjoma optimizēta programmas versija, kura ir paredzēta lietotāja lietošanai. Programmētāji izmanto tādu versiju, lai nodotu pabeigto programmu klientam.

Breakpoints

Izmantojot taustiņu **F9** (Toggle Breakpoint) var apzīmēt jebkuru programmas rindu ka pārtraukumpunktu (breakpoint), tad kad programmas darba kārtība nonāks līdz šai rindai, programmas izpildīšana būs pārtraukta un programma pāries atkļūdošanas režīmā. Lai atceltu pārtraukumpunktu var vēlreiz uzspiest taustiņu **F9** vai taustiņu kombināciju **Ctrl+Shift+F9** lai atceltu visus pārtraukumpunktus programmā.

F9	Toggle Breakpoint	Uzstādīt / atcelt pārtraukumpunktu
Ctrl+Shift+F9	Delete All Breakpoints	Dzēst visus pārtraukumpunktus programmā
Alt+F9, C	Breakpoint Condition	Noteikt pārtraukumpunkta iedarbināšanas nosacījumu
	Breakpoint Action	Noteikt pārtraukumpunkta darbības
F5	Continue	Turpināt līdz nākamajam pārtraukumpunktam
Shift+F5	Stop Debugging	Apstādināt programmu
Ctrl+Shift+F5	Restart Program	Palaist programmu no sākuma
Ctrl+F5	Start Without Debugging	Palaist programmu, nereaģējot uz pārtraukumpunktus

Watch

Atkļūdošanas laikā var uzzināt vai pat izmainīt mainīgā vērtību. Lai uzzinātu mainīgā vērtību var pārvietot un aizkavēt peles kursoru pie mainīgā vārda jebkura programmas vietā, blakus kursoram parādīsies mainīgā nozīme. Arī var izmantot taustiņu kombināciju **Shift+F9** (Quick Watch), pēc tam atsevišķā logā var pārbaudīt un izmainīt jebkurā mainīgā vērtību.

Shift+F9	Quick Watch	Parādīt mainīgā vērtību atsevišķā logā
	Pin	Rādīt mainīgā vērtību programmas kodā blakus tā vārda
	Add Watch	Rādīt mainīgā vērtību uzraudzīšanas logā

Trace

Trasēšana – tā ir programmas izpildīšana "pa vienai rindai", kas ļauj izsekot programmas darba kārtību. Trasēšanai ir jāizmanto taustiņi **F10** (Step Into) un **F11** (Step Over). Taustiņš **F11** ļauj izpildīt vienu tekošo programmas rindu un pāriet pie nākamas rindas, savukārt, ja tekoša rinda satur funkcijas izsaukumu, taustiņš **F10** ļauj ienākt funkcijas iekšā un trasēt tā darbu.

F10	Step Into	Ienākt uz funkciju
F11	Step Over	Izpildīt tekošo rindu
Shift+F11	Step Out	Izpildīt līdz funkcijas beigām
Ctrl+F10	Run To Cursor	Izpildīt līdz rindas ar kursoru
F5	Continue	Turpināt izpildīšanu nepārtrauktā režīmā
Shift+F5	Stop Debugging	Apstādināt programmu
Ctrl+Shift+F5	Restart Program	Palaist programmu no sākuma

Nosacīta kompilācija

Programmas kodā var apzīmēt fragmentus, kas būs izpildītie tikai atkļūdošanas versijā. Izlaišanas versijas kompilācijas laikā šie fragmenti būs automātiski izslēgti. To panāk, izmantojot priekšprocesora direktīvu `#if...#endif`

Izpildīt tikai atkļūdošanās versijā

```
#if _DEBUG
    std::cout << "This is debug version\n";
#endif
```

Izpildīt dažādus fragmentus atkļūdošanās un izlaišanās versijās

```
#if _DEBUG
     std::cout << "This is debug version\n";
#else
     std::cout << "This is release version\n";
#endif
```

Izpildīt tikai izlaišanās versijā

```
#if ! _DEBUG
    std::cout << "This is release version\n";
#endif
```