

Verifikacija softvera - Pacman

Projekat za analizu: Pacman

Adresa projekta: <https://github.com/MATF-RS20/RS030-pacman>

Alati za verifikaciju softvera će biti primenjeni commit sa hešom
22218d726c58f913ca10168f73bbd15aa47ceb98

Ovaj projekat predstavlja video-igru po uzoru na poznatu igru Pacman

Projekat je implementiran uz pomoć C++ i Qt okruženja.

1. Alat - Valgrind Memcheck

Pre nego što se pokrene alat Memcheck, u Qt okruženje dodajemo linije koda -g i -O0 kako bi imali dobro korelisane linije na kojima su pronađene greške i linije koda u izvornom kodu, kao i da bi se eliminisale optimizacije koje potencijalno mogu zamaskirati neke probleme.

```
5 CONFIG += c++17
6
7 # Set optimization level to O0
8 QMAKE_CXXFLAGS_DEBUG += -O0
9
10 # Add -g option for debugging symbols
11 QMAKE_CXXFLAGS_DEBUG += -g
12
```

Nadalje, pokrećemo alat pozivom funkcije:

```
valgrind --tool=memcheck --leak-check=full --show-leak-kinds=all --log-  
file="Memcheck_report" --track-origins=yes ./qt
```

Prilikom završetka rada programa, dobijamo fajl koji sadrži informacije o memoriji:

```
LEAK SUMMARY:
  definitely lost: 5,960 bytes in 240 blocks
  indirectly lost: 7,672 bytes in 239 blocks
  possibly lost: 768 bytes in 2 blocks
  still reachable: 2,887,377 bytes in 30,417 blocks
  suppressed: 0 bytes in 0 blocks
```

Ovde se vidi da postoji jako mnogo propusta u pravljenju programa, ali kao primer, uzet je jedan problem sa alokacijom memorije:

```
8 bytes in 1 blocks are definitely lost in loss record 333 of 9,378
at 0x4849013: operator new(unsigned long) (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
by 0x124FCB: Game::gameOver(QString) (game.cpp:246)
```

Dakle, u liniji 246 u funkciji gameOver, klase Game, dolazi do curenja memorije.

```
246 | highScores[9] = std::pair<QString*,int>(new QString(player), sk);
247 | QString *whichPlayer = new QString(&player + QString::number(howManyGames));
248 | howManyGames++;
249 | highScores[9] = std::pair<QString*,int>(whichPlayer, sk);
```

Promenljiva highScores je par koji sadrži pokazivač na objekat QString i int. Generalno nije dobra praksa koristiti pokazivače za jednostavne tipove kao što je QString, a u ovom slučaju memorija koju highScores[9] čuva nije dealocirana pre nove dodele. Ovaj kod bi trebao da izgleda na primer, ovako:

```
246 | if (highScores[9].first != nullptr) {
247 |     delete highScores[9].first;
248 | }
249 | QString whichPlayer = player + QString::number(howManyGames);
250 | highScores[9] = std::pair<QString*, int>(new QString(whichPlayer), sk);
251 | howManyGames++;
```

2. Alat - Valgrind Callgrind

Alat se pokreće pozivom funkcije:

```
valgrind --tool=callgrind --cache-sim=yes --branch-sim=yes ./qt
```

Vizuelnu reprezentaciju poziva funkcija dobijamo pokretanjem alata za prikaz komandom:

```
kcachegrind callgrind.out.26203
```

