

Analiza projekta Alat za analitiku

Projekat iz Verifikacije softvera

Jelisaveta Gavrilović 1028/2024

9. oktobar 2025.

Sadržaj

1	Uvod	2
2	Valgrind	2
2.1	Instalacija	3
2.2	Memcheck	3
2.2.1	Pokretanje alata	4
2.2.2	Rezultati analize	5
2.3	Massif	6
2.3.1	Pokretanje alata	6
2.3.2	Rezultati analize	7
3	Clang-Format	8
3.1	Instalacija	9
3.2	Pokretanje alata	9
3.3	Analiza rezultata	10
4	Cppcheck	10
4.1	Instalacija	11
4.2	Pokretanje alata	11
4.3	Analiza rezultata	11

1 Uvod

Projekat koji je predmet ove analize je [Alat za analitiku](#). Analiza je rađena nad granom `main` i commit-om [2f803aed](#).

Alat za analitiku je softverski alat koji omogućava generisanje izveštaja, statistika i vizualizacija nad podacima. Podaci se učitavaju iz datoteka i, po potrebi, vrši se konverzija tipova. Korisnik može da prikazuje različite grafike, uključujući raspodele, bar plot-ove i analize zavisnosti podataka, kao i da štampa ili čuva izveštaje. Ovaj alat se može posmatrati kao mini verzija alata poput PowerBI-ja, namenjena analizi i vizualizaciji podataka.

Cilj ove analize je da se kroz korišćenje alata za verifikaciju softvera identifikuju potencijalni problemi u kodu i pruže smernice za optimizaciju, poboljšanje stila i standardizaciju koda. Za ovu svrhu su korišćeni sledeći alati:

- **Valgrind – Memcheck** za detekciju curenja memorije,
- **Valgrind – Massif** za praćenje potrošnje memorije,
- **Clang-Format** za proveru i automatsko formatiranje koda,
- **Cppcheck** za statičku analizu koda.

2 Valgrind

Valgrind je okvir (*framework*) za dinamičku analizu programa koji omogućava otkrivanje širokog spektra problema u radu sa memorijom, optimizaciju performansi i praćenje resursa tokom izvršavanja aplikacije. U osnovi, Valgrind radi tako što izvršava program unutar virtuelne mašine koja nadgleda svaku instrukciju i pristup memoriji, čime omogućava detaljno praćenje ponašanja programa na nivou bajta.

Valgrind ne zahteva izmene u izvornom kodu — pokreće se nad već kompajliranim izvršnim fajlom. Zbog toga je posebno koristan u fazi verifikacije i testiranja softvera, jer može da detektuje:

- curenja memorije (*memory leaks*),
- pristup oslobođenoj ili neinicijalizovanoj memoriji,
- nevažeće pokazivače i prepisivanje memorijskih granica,
- nepotpuno oslobađanje resursa (npr. otvoreni fajlovi ili deskriptori),
- neefikasnu ili prekomernu upotrebu memorije.

Valgrind se sastoji od više specijalizovanih alata, koji se biraju parametrom **-tool=** prilikom pokretanja. Najznačajniji među njima su:

- **Memcheck** – detektuje greške u radu sa memorijom (neinicijalizovane promenljive, curenja memorije, nevažeći pokazivači i slično)
- **Massif** – meri i prikazuje potrošnju memorije tokom izvršavanja programa, omogućavajući profilisanje i analizu performansi
- **Cachegrind** – analizira keš memoriju i procese grananja radi optimizacije performansi procesora
- **Callgrind** – prati tok poziva funkcija i meri vreme izvršavanja svake funkcije.
- **Helgrind** i **DRD** – detektuju probleme u višenitnim (multithreaded) aplikacijama

2.1 Instalacija

Na Linux sistemima, Valgrind se može jednostavno instalirati pokretanjem komande u terminalu:

```
sudo apt install valgrind
```

Nakon instalacije, dostupna je osnovna komanda:

```
valgrind --tool=<alat> <izvršni_fajl>
```

gde se, u zavisnosti od izabranog alata, analiziraju različiti aspekti rada programa.

U ovom projektu korišćena su dva Valgrind alata:

- **Memcheck** – za detekciju grešaka i curenja memorije.
- **Massif** – za praćenje i vizuelizaciju potrošnje memorije tokom izvršavanja.

Njihovi rezultati i analiza predstavljeni su u narednim odeljcima.

2.2 Memcheck

Memcheck je alat za dinamičku analizu memorije koji omogućava detekciju grešaka u radu sa memorijom u C i C++ programima. Koristi se za pronađenje:

- curenja memorije (memory leaks),

- neinicijalizovanih promenljivih,
- nevalidnih pristupa memoriji (segmentation faults),
- pogrešnih oslobođanja memorije.

Memcheck funkcioniše tako što izvršava program u posebnom okruženju koje prati svaku operaciju nad memorijom, beleži potencijalne probleme i prikazuje detaljne izveštaje o njima. Na ovaj način se mogu uočiti problemi koji nisu vidljivi prilikom standardnog testiranja.

2.2.1 Pokretanje alata

Za pokretanje alata kreirana je skripta `run_memcheck.sh`, koja automatski pokreće proces izvršavanja Memcheck-a nad glavnim izvršnim fajlom projekta. Da bismo pokrenuli skriptu, potrebno je da se pozicioniramo u direktorijum `valgrind/memcheck/` i izvršimo sledeće komande:

```
chmod +x run_memcheck.sh
./run_memcheck.sh
```

Komanda kojom pokrećemo Memcheck u skripti je sledeća:

```
valgrind --tool=memcheck \
    --leak-check=summary \
    --track-origins=yes \
    --suppressions="suppressions.sup" \
    --log-file="results.txt" \
    ../../alat-za-analitiku/alatZaAnalitiku/build/
Desktop_Qt_6_9_3-Debug/alatZaAnalitiku
```

Objašnjenje parametara:

- `-tool=memcheck` – pokreće Memcheck alat
- `-leak-check=summary` – prikazuje sažetak curenja memorije
- `-track-origins=yes` – pokušava da prati "poreklo" neinicijalizovanih vrednosti
- `-suppressions=suppressions.sup` – koristi fajl sa izuzecima za poznate, ignorisane greške - greške koje dolaze iz sistemskih, Qt biblioteka i na taj način nam omogućava da pratimo samo naš kod
- `-log-file="results.txt"` – zapisuje izlaz Memcheck-a u fajl.
- poslednji argument je putanja do izvršnog fajla aplikacije

Rezultati analize se čuvaju u fajlu `results.txt`, koji se nalazi u istom direktorijumu. Ovaj fajl sadrži detaljan izveštaj o svim detektovanim curenjima memorije, neinicijalizovanim vrednostima i drugim problemima koji su uočeni tokom izvršavanja programa.

2.2.2 Rezultati analize

Analiza je pokazala da projekat Alat za analitiku sadrži određene probleme u radu sa memorijom. Ključni delovi iz izveštaja su sledeći:

- **Use of uninitialised value** – u funkciji `DynamicUI::openPlotWindow()` (linija 74, fajl `dynamicui.cpp`) koristi se neinicijalizovana vrednost, čije poreklo vodi do alokacije u funkciji `FileImportWindow::dropEvent()` (linija 49, `fileimportwindow.cpp`).
- **Curenja memorije:**
 - `definitely lost: 20,016 bytes in 16 blocks` – predstavlja stvarno curenje memorije, gde je memorija alocirana, ali je pokazivač izgubljen i više joj se ne može pristupiti. Ovakvi gubici obično nastaju kada se dinamički alocirani objekti ne oslobođe nakon upotrebe.
 - `indirectly lost: 492,354 bytes in 3,072 blocks` – odnosi se na objekte koji su zavisni od drugih alokacija. Kada glavni pokazivač bude izgubljen, svi pokazivači koji ukazuju na njegove delove postaju indirektno izgubljeni.
 - `possibly lost: 32,947 bytes in 233 blocks` – potencijalni gubici memorije koji se najčešće javljaju kod složenih struktura (npr. pokazivači unutar objekata), gde Valgrind ne može sa sigurnošću da utvrdi pristupnost memoriji.
 - `still reachable: 15,593,825 bytes in 59,018 blocks` – memorija koja nije oslobođena pre završetka programa, ali je i dalje dostupna. U kontekstu Qt aplikacija ovo je uobičajeno ponašanje i ne mora nužno predstavljati grešku, jer Qt često zadržava određene resurse do potpunog gašenja aplikacije.

Statistika potrošnje memorije:

- **Ukupan broj alokacija:** 1,699,269
- **Ukupan broj oslobođenih blokova:** 1,633,089
- **Ukupno alocirano:** 426,406,236 bajtova (\approx 426 MB)

- **Memorija u upotrebi pri izlasku programa:** 16,590,246 bajtova u 66,180 blokova

Na kraju izveštaja zabeleženo je: **ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 1013 from 732)**, što znači da je Memcheck detektovao jednu stvarnu grešku koja potiče iz korisničkog koda, dok su ostale greške detektovane, ali potisnute (“suppressed”) jer potiču iz eksternih biblioteka, pretežno iz samog Qt okvira.

Važno je napomenuti da, iako je Memcheck u ovom izvršavanju uspešno završio analizu, primećeno je da se program prilikom ponovljenih pokretanja ponekad "puca" usled pristupa neinicijalizovanoj memoriji (**SIGSEGV**).

2.3 Massif

Massif je alat za profilisanje memorije u C/C++ aplikacijama. Njegova primarna funkcija je praćenje upotrebe heap memorije tokom izvršavanja programa. Massif beleži koliko memorije aplikacija koristi u različitim trenucima i omogućava identifikaciju mesta u kodu gde dolazi do prekomerne alokacije ili curenja memorije.

Alat je posebno koristan za:

- otkrivanje potencijalnih problema sa alokacijom memorije,
- analizu performansi i optimizaciju memorijskih resursa,
- detekciju skrivenih grešaka u upravljanju memorijom, koje se ponekad ne manifestuju pri standardnom izvršavanju programa.

2.3.1 Pokretanje alata

U ovom projektu, Massif se pokreće kroz skriptu **run_massif.sh** koja se nađe u direktorijumu **valgrind/massif/**. Skripta obezbeđuje pravilnu konfiguraciju i parametre i pokreće se komandama:

```
chmod +x run_massif.sh  
./run_massif.sh
```

Komanda koja se koristi u skripti za pokretanje alata:

```
valgrind --tool=massif --stacks=no --time-unit=ms --threshold=5.0 \  
--detailed-freq=200 --massif-out-file=massif.out ./alatZaAnalitiku
```

Parametri znače:

- **-tool=massif** – pokreće Massif modul Valgrind-a
- **-stacks=no** – isključuje praćenje stack memorije
- **-time-unit=ms** – koristi milisekunde kao jedinicu vremena
- **-threshold=5.0** – beleži promene memorije veće od 5%
- **-detailed-freq=200** – detaljna frekvencija prikupljanja podataka
- **-massif-out-file=massif.out.<PID>** – ime izlaznog fajla sa profilom memorije

Rezultat izvršavanja skripte je profil memorije koji nam može otkriti funkcije ili objekte koji najviše troše memoriju. Izlazni fajl se nalazi u direktorijumu **massif_out** i može se pregledati kroz terminal komandnom linijom:

```
ms_print maassif_out/massif.out.<PID>
```

ili grafičkim prikazom pomoću Massif Visualizer alata:

```
massif-visualizer maassif_out/massif.out.<PID>
```

2.3.2 Rezultati analize

Tokom pokretanja aplikacije pod Massif-om, program je prekidao izvršavanje (pucao) sa signalom **SIGSEGV** zbog pokušaja pristupa neinicijalizovanom pokazivaču. Konkretno, u funkciji **DynamicUI::openPlotWindow()** linija 74:

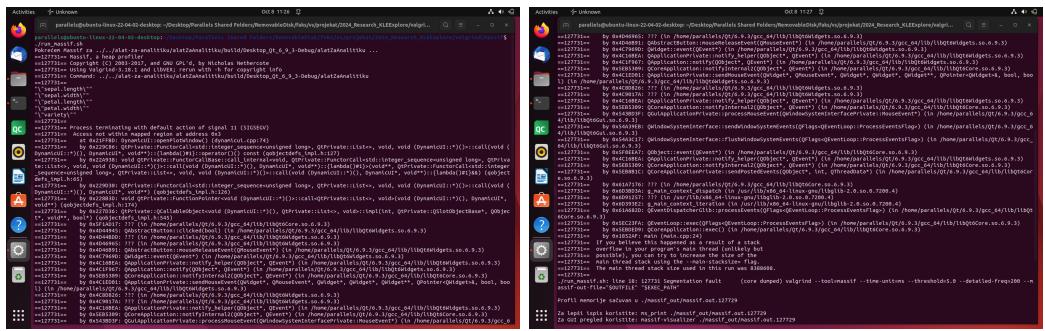
```
if(p) delete p;
```

pokazivač **p** nije inicijalizovan na **nullptr**, što dovodi do nedefinisanog ponašanja i padu programa. Sličan problem se javlja u funkciji **openStatWindow()** sa pokazivačem **s**.

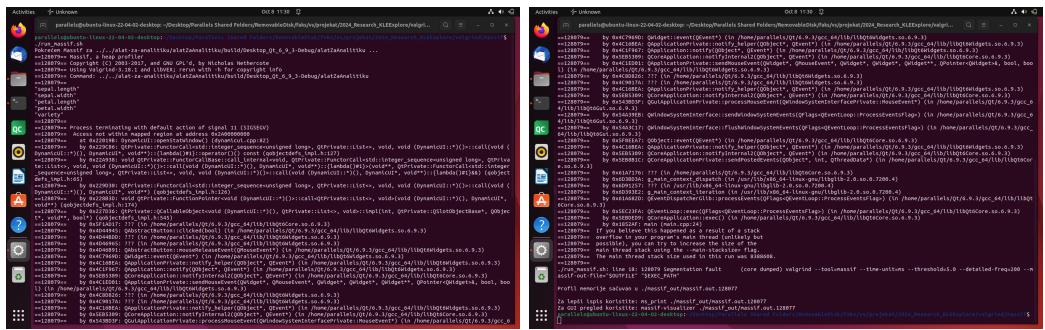
Zbog toga što program puca pre nego što normalno završi izvršavanje, Massif beleži nepotpun profil memorije. Izlaz alata stoga nije pouzdan za analizu stvarnog zauzeća memorije ili optimizaciju.

Ovaj problem ne mora da se manifestuje pri standardnom izvođenju programa jer:

- raspored memorije u normalnom izvršavanju ponekad slučajno stavlja neinicijalizovane pokazivače na adresu koja ne izaziva segfault,



Slika 1: Pokretanje aplikacije i SIGSEGV greška pri funkciji `openPlotWindow()`.



Slika 2: Pokretanje aplikacije i SIGSEGV greška pri funkciji `openStatWindow()`.

- Qt framework automatski "menadžuje" parent-child odnos widgeta, što u nekim slučajevima sprečava direktni pad,
- brzina izvršavanja i stanje heap memorije u realnom runtime okruženju utiču na to da se greška ne vidi uvek.

Međutim, ovaj alat strogo proverava sve operacije sa memorijom i odmah detektuje pristup neinicijalizovanim ili već obrisanim pokazivačima, zbog čega se crash dosledno javlja prilikom analize.

3 Clang-Format

Clang-Format je alat koji automatski formatira izvorni kod napisan u jezicima C, C++, Java, JavaScript i drugim, prema definisanim pravilima stil. Njegov glavni cilj je da obezbedi konzistentan izgled koda nezavisno od pro-

gramera, čime se povećava čitljivost, održavanje i preglednost projekata.

Formatiranje se vrši prema pravilima definisanim u konfiguracionom fajlu `.clang-format`, koji može da nasleđuje neki od postojećih stilova (*LLVM, Google, Mozilla, Microsoft*, itd.), a zatim se po potrebi dalje prilagođava.

3.1 Instalacija

Na Linux sistemima alat se može instalirati komandom:

```
sudo apt install clang-format
```

3.2 Pokretanje alata

Za potrebe projekta kreirana je skripta `run_clangFormat.sh` koja automatskuje proces formatiranja svih `.cpp` i `.h` fajlova u okviru projekta. Skripta pretražuje sve relevantne fajlove (isključujući `tests`, `build` i `CMakeFiles` direktorijume) i za svaki od njih primenjuje `clang-format` prema unapred definisanom stilu.

Pokretanjem komande:

```
clang-format -style=LLVM -dump-config > styleFile
```

generišemo fajl `styleFile` koji sadrži podrazumevana pravila LLVM stila, što nam omogućava pregled i selekciju parametara koje možemo da promenimo. Na sličan način je moguće generisati fajl i za druge stilove.

Za formatiranje koda korišćen je stil zasnovan na LLVM šablonu, uz nekoliko prilagođenih parametara definisanih direktno u skripti.

```
clang-format -style="{ \
    BasedOnStyle: LLVM, \
    IndentWidth: 4, \
    TabWidth: 4, \
    AlignConsecutiveAssignments: true, \
    AlignEscapedNewlines: Left, \
    AllowShortBlocksOnASingleLine: true, \
    AllowShortCaseLabelsOnASingleLine: true, \
    AllowShortFunctionsOnASingleLine: Inline, \
    AllowShortLambdasOnASingleLine: All, \
    BreakBeforeBraces: Attach, \
    MaxEmptyLinesToKeep: 2, \
    ColumnLimit: 150}"
```

Ovi parametri nam omogućavaju:

- `IndentWidth`: 4 i `TabWidth`: 4 – uvlačenje od četiri razmaka radi bolje čitljivosti koda,
- `AlignConsecutiveAssignments`: `true` – poravnava uzastopne dodele radi preglednosti,
- `AllowShortFunctionsOnASingleLine`: `Inline` i `AllowShortLambdasOnASingleLine`: `All` – dozvoljava kraće funkcije i lambda izraze u jednom redu,
- `BreakBeforeBraces`: `Attach` – otvarajuća vitičasta zagrada se nalazi u istom redu kao i naredba,
- `ColumnLimit`: 150 – povećava maksimalnu dužinu reda radi smanjenja nepotrebnih preloma u kompleksnim izrazima.

Ovakav pristup je bio neophodan jer u okviru projekta već postoji `.clang-format` fajl zasnovan na *Microsoft* stilu. Da bi se izbeglo njegovo automatsko nasleđivanje, željeni stil je definisan direktno unutar skripte (`inline`), čime je osigurano da se uvek koristi *LLVM*-bazirani stil sa prilagođenim parametrima.

Pokretanje skripte se vrši sledećim komandama iz direktorijuma `clang-format`:

```
chmod +x run_clangFormat.sh  
./run_clangFormat.sh
```

3.3 Analiza rezultata

Skripta kreira direktorijum `formattedFiles` u koji se smeštaju svi fajlovi čiji format nije bio u skladu sa zadatim pravilima.

4 Cppcheck

Cppcheck je statički analizator koda za programske jezike C i C++. Njegova osnovna funkcija je da analizira izvorni kod bez njegovog izvršavanja i identificuje potencijalne greške i rizične konstrukcije. Alat može detektovati probleme kao što su neinicijalizovane promenljive, logičke greške, curenje memorije, neoptimalni izrazi i druge potencijalne izvore bug-ova. Za razliku od kompjajlera, koji prijavljuje samo sintaksne greške i konstrukcije koje sprečavaju kompilaciju, Cppcheck upozorava na probleme koji mogu dovesti do neочекivanog ponašanja programa tokom izvršavanja. Korišćenjem Cppcheck-

a, programeri mogu unaprediti kvalitet, pouzdanost i sigurnost svog koda, smanjujući rizik od kritičnih grešaka u produkcionom okruženju.

4.1 Instalacija

Na Linux sistemima, Cppcheck se može instalirati komandom:

```
sudo apt install cppcheck
```

4.2 Pokretanje alata

U ovom projektu, Cppcheck je pokrenut kroz skriptu `run_cppcheck.sh` koja se nalazi u direktorijumu `cppcheck.sh`. Skripta automatski generiše izveštaje u tekstualnom i HTML formatu.

Pokretanje:

```
chmod +x run_cppcheck.sh  
./run_cppcheck.sh
```

Skripta uključuje sledeće korake:

- Definisanje putanje do projekta i direktorijuma za izveštaje.
- Pokretanje `cppcheck` sa parametrima:
 - `-enable=all` — omogućava sve vrste provera,
 - `-inconclusive` — uključuje i moguće neodređene greške,
 - `-std=c++17` — koristi C++17 standard,
 - `-I` — dodaje include direktorijume (Qt biblioteke),
 - `-D` — definiše makro (QT_CORE_LIB),
 - `-suppress=missingIncludeSystem` — ignoriše greške vezane za sistemske include fajlove.
- Generisanje XML fajla za kasniju konverziju u HTML izveštaj.
- Kreiranje HTML izveštaja pomoću `cppcheck-htmlreport`.

4.3 Analiza rezultata

Svi rezultati analize se nalaze u direktorijumu `cppcheck-report`. Mogu se pregledati u dva formata:

- **Tekstualni izveštaj:** `cppcheck-output.txt` sadrži listu svih detektovanih problema i njihov opis.
- **HTML izveštaj:** `index.html` u direktorijumu `cppcheck-report` omogućava interaktivni pregled, sa lakšim filtriranjem i navigacijom po fajlovima.

`index.html` Ovaj izveštaj pruža detaljan prikaz svih detektovanih problema po fajlovima i linijama koda. Svaka greška ili upozorenje je opisano zajedno sa vrstom problema i preporukom za ispravku. `index.html` omogućava lako filtriranje i navigaciju kroz kod, što programerima omogućava brzo pronađenje kritičnih delova projekta.

`stats.html` Ovaj izveštaj sadrži sumarne statistike analize. Prikazuje broj problema po tipu (npr. greške, upozorenja, moguće greške) i njihovu ozbiljnost. Statistički prikaz omogućava brz uvid u opšte stanje koda, identifikovanje najproblematičnijih modula i planiranje prioriteta za ispravke.

Kombinovanjem `index.html` i `stats.html`, programeri dobijaju potpun pregled kvaliteta koda i mogu efikasno pratiti napredak u otklanjanju grešaka.

...

```

2024Research_KLEExplorers 2024Research_KLEExplorers
plotwindow.cpp plotwindow.cpp
plotwindow.cpp:162,163: warning: unused variable 'ui' [-Wunused-variable]
  162   ui->buttonBox->button(QDialogButtonBox::Ok)->setEnabled(false);
  163   ui->buttonBox->button(QDialogButtonBox::Cancel)->setEnabled(true);
  164
  165   if (arg1 == "Scatter plot")
  166   {
  167     setupTwo();
  168   }
  169   else if (arg1 == "Line plot")
  170   {
  171     setupTwo();
  172   }
  173   else
  174   {
  175     setupOneAll();
  176   }
  177
  178   ui->buttonMakePlot->show();
  179
  180   QChart *plotWindow = getChartFromConfig();
  181
  182   QString x = ui->cX->currentText();
  183   QString y = ui->cY->currentText();
  184   QString type = ui->cType->currentText();
  185   QString stat = ui->cStat->currentText();
  186   QColor color = currentColor();
  187   QFont font = ui->font;
  188
  189   QChart chart = returnPlot(type, data, x, y, stat, color);
  190
  191   // ui->lineEdit->text().length() > MAX_LENGTH
  192
  193   if (ui->lineEdit->text().length() > MAX_LENGTH)
  194   {
  195     QMessageBox msgBox;
  196     msgBox.setWindowIcon(QIcon(":/warning"));
  197     msgBox.setWindowTitle("Character Limit Exceeded");
  198     msgBox.setDetailedText("The maximum allowed length is 15 characters.");
  199     msgBox.setStandardButtons(QMessageBox::Ok);
  200     msgBox.exec();
  201   }
  202
  203   chart->setTitle(ui->lineEdit->text());
  204
  205   QFont titleFont;
  206   titleFont.setFamily("Arial");
  207
  208   chart->formatTitles();
  209
  210   chart->setTitle(ui->lineEdit->text());
  211
  212   chart->setLabels();
  213
  214   chart->setXLabels();
  215
  216   chart->setYLabels();
  217
  218   chart->setXAxis();
  219
  220   chart->setYAxis();
  221
  222   chart->setLegend();
  223
  224   chart->setSeries();
  225
  226   chart->setPlotArea();
  227
  228   chart->setGrid();
  229
  230   chart->setAxes();
  231
  232   chart->setLabels();
  233
  234   chart->setXLabels();
  235
  236   chart->setYLabels();
  237
  238   chart->setXAxis();
  239
  240   chart->setYAxis();
  241
  242   chart->setLegend();
  243
  244   chart->setSeries();
  245
  246   chart->setPlotArea();
  247
  248   chart->setGrid();
  249
  250   chart->setAxes();
  251
  252   chart->setLabels();
  253
  254   chart->setXLabels();
  255
  256   chart->setYLabels();
  257
  258   chart->setXAxis();
  259
  260   chart->setYAxis();
  261
  262   chart->setLegend();
  263
  264   chart->setSeries();
  265
  266   chart->setPlotArea();
  267
  268   chart->setGrid();
  269
  270   chart->setAxes();
  271
  272   chart->setLabels();
  273
  274   chart->setXLabels();
  275
  276   chart->setYLabels();
  277
  278   chart->setXAxis();
  279
  280   chart->setYAxis();
  281
  282   chart->setLegend();
  283
  284   chart->setSeries();
  285
  286   chart->setPlotArea();
  287
  288   chart->setGrid();
  289
  290   chart->setAxes();
  291
  292   chart->setLabels();
  293
  294   chart->setXLabels();
  295
  296   chart->setYLabels();
  297
  298   chart->setXAxis();
  299
  300   chart->setYAxis();
  301
  302   chart->setLegend();
  303
  304   chart->setSeries();
  305
  306   chart->setPlotArea();
  307
  308   chart->setGrid();
  309
  310   chart->setAxes();
  311
  312   chart->setLabels();
  313
  314   chart->setXLabels();
  315
  316   chart->setYLabels();
  317
  318   chart->setXAxis();
  319
  320   chart->setYAxis();
  321
  322   chart->setLegend();
  323
  324   chart->setSeries();
  325
  326   chart->setPlotArea();
  327
  328   chart->setGrid();
  329
  330   chart->setAxes();
  331
  332   chart->setLabels();
  333
  334   chart->setXLabels();
  335
  336   chart->setYLabels();
  337
  338   chart->setXAxis();
  339
  340   chart->setYAxis();
  341
  342   chart->setLegend();
  343
  344   chart->setSeries();
  345
  346   chart->setPlotArea();
  347
  348   chart->setGrid();
  349
  350   chart->setAxes();
  351
  352   chart->setLabels();
  353
  354   chart->setXLabels();
  355
  356   chart->setYLabels();
  357
  358   chart->setXAxis();
  359
  360   chart->setYAxis();
  361
  362   chart->setLegend();
  363
  364   chart->setSeries();
  365
  366   chart->setPlotArea();
  367
  368   chart->setGrid();
  369
  370   chart->setAxes();
  371
  372   chart->setLabels();
  373
  374   chart->setXLabels();
  375
  376   chart->setYLabels();
  377
  378   chart->setXAxis();
  379
  380   chart->setYAxis();
  381
  382   chart->setLegend();
  383
  384   chart->setSeries();
  385
  386   chart->setPlotArea();
  387
  388   chart->setGrid();
  389
  390   chart->setAxes();
  391
  392   chart->setLabels();
  393
  394   chart->setXLabels();
  395
  396   chart->setYLabels();
  397
  398   chart->setXAxis();
  399
  400   chart->setYAxis();
  401
  402   chart->setLegend();
  403
  404   chart->setSeries();
  405
  406   chart->setPlotArea();
  407
  408   chart->setGrid();
  409
  410   chart->setAxes();
  411
  412   chart->setLabels();
  413
  414   chart->setXLabels();
  415
  416   chart->setYLabels();
  417
  418   chart->setXAxis();
  419
  420   chart->setYAxis();
  421
  422   chart->setLegend();
  423
  424   chart->setSeries();
  425
  426   chart->setPlotArea();
  427
  428   chart->setGrid();
  429
  430   chart->setAxes();
  431
  432   chart->setLabels();
  433
  434   chart->setXLabels();
  435
  436   chart->setYLabels();
  437
  438   chart->setXAxis();
  439
  440   chart->setYAxis();
  441
  442   chart->setLegend();
  443
  444   chart->setSeries();
  445
  446   chart->setPlotArea();
  447
  448   chart->setGrid();
  449
  450   chart->setAxes();
  451
  452   chart->setLabels();
  453
  454   chart->setXLabels();
  455
  456   chart->setYLabels();
  457
  458   chart->setXAxis();
  459
  460   chart->setYAxis();
  461
  462   chart->setLegend();
  463
  464   chart->setSeries();
  465
  466   chart->setPlotArea();
  467
  468   chart->setGrid();
  469
  470   chart->setAxes();
  471
  472   chart->setLabels();
  473
  474   chart->setXLabels();
  475
  476   chart->setYLabels();
  477
  478   chart->setXAxis();
  479
  480   chart->setYAxis();
  481
  482   chart->setLegend();
  483
  484   chart->setSeries();
  485
  486   chart->setPlotArea();
  487
  488   chart->setGrid();
  489
  490   chart->setAxes();
  491
  492   chart->setLabels();
  493
  494   chart->setXLabels();
  495
  496   chart->setYLabels();
  497
  498   chart->setXAxis();
  499
  500   chart->setYAxis();
  501
  502   chart->setLegend();
  503
  504   chart->setSeries();
  505
  506   chart->setPlotArea();
  507
  508   chart->setGrid();
  509
  510   chart->setAxes();
  511
  512   chart->setLabels();
  513
  514   chart->setXLabels();
  515
  516   chart->setYLabels();
  517
  518   chart->setXAxis();
  519
  520   chart->setYAxis();
  521
  522   chart->setLegend();
  523
  524   chart->setSeries();
  525
  526   chart->setPlotArea();
  527
  528   chart->setGrid();
  529
  530   chart->setAxes();
  531
  532   chart->setLabels();
  533
  534   chart->setXLabels();
  535
  536   chart->setYLabels();
  537
  538   chart->setXAxis();
  539
  540   chart->setYAxis();
  541
  542   chart->setLegend();
  543
  544   chart->setSeries();
  545
  546   chart->setPlotArea();
  547
  548   chart->setGrid();
  549
  550   chart->setAxes();
  551
  552   chart->setLabels();
  553
  554   chart->setXLabels();
  555
  556   chart->setYLabels();
  557
  558   chart->setXAxis();
  559
  560   chart->setYAxis();
  561
  562   chart->setLegend();
  563
  564   chart->setSeries();
  565
  566   chart->setPlotArea();
  567
  568   chart->setGrid();
  569
  570   chart->setAxes();
  571
  572   chart->setLabels();
  573
  574   chart->setXLabels();
  575
  576   chart->setYLabels();
  577
  578   chart->setXAxis();
  579
  580   chart->setYAxis();
  581
  582   chart->setLegend();
  583
  584   chart->setSeries();
  585
  586   chart->setPlotArea();
  587
  588   chart->setGrid();
  589
  590   chart->setAxes();
  591
  592   chart->setLabels();
  593
  594   chart->setXLabels();
  595
  596   chart->setYLabels();
  597
  598   chart->setXAxis();
  599
  600   chart->setYAxis();
  601
  602   chart->setLegend();
  603
  604   chart->setSeries();
  605
  606   chart->setPlotArea();
  607
  608   chart->setGrid();
  609
  610   chart->setAxes();
  611
  612   chart->setLabels();
  613
  614   chart->setXLabels();
  615
  616   chart->setYLabels();
  617
  618   chart->setXAxis();
  619
  620   chart->setYAxis();
  621
  622   chart->setLegend();
  623
  624   chart->setSeries();
  625
  626   chart->setPlotArea();
  627
  628   chart->setGrid();
  629
  630   chart->setAxes();
  631
  632   chart->setLabels();
  633
  634   chart->setXLabels();
  635
  636   chart->setYLabels();
  637
  638   chart->setXAxis();
  639
  640   chart->setYAxis();
  641
  642   chart->setLegend();
  643
  644   chart->setSeries();
  645
  646   chart->setPlotArea();
  647
  648   chart->setGrid();
  649
  650   chart->setAxes();
  651
  652   chart->setLabels();
  653
  654   chart->setXLabels();
  655
  656   chart->setYLabels();
  657
  658   chart->setXAxis();
  659
  660   chart->setYAxis();
  661
  662   chart->setLegend();
  663
  664   chart->setSeries();
  665
  666   chart->setPlotArea();
  667
  668   chart->setGrid();
  669
  670   chart->setAxes();
  671
  672   chart->setLabels();
  673
  674   chart->setXLabels();
  675
  676   chart->setYLabels();
  677
  678   chart->setXAxis();
  679
  680   chart->setYAxis();
  681
  682   chart->setLegend();
  683
  684   chart->setSeries();
  685
  686   chart->setPlotArea();
  687
  688   chart->setGrid();
  689
  690   chart->setAxes();
  691
  692   chart->setLabels();
  693
  694   chart->setXLabels();
  695
  696   chart->setYLabels();
  697
  698   chart->setXAxis();
  699
  700   chart->setYAxis();
  701
  702   chart->setLegend();
  703
  704   chart->setSeries();
  705
  706   chart->setPlotArea();
  707
  708   chart->setGrid();
  709
  710   chart->setAxes();
  711
  712   chart->setLabels();
  713
  714   chart->setXLabels();
  715
  716   chart->setYLabels();
  717
  718   chart->setXAxis();
  719
  720   chart->setYAxis();
  721
  722   chart->setLegend();
  723
  724   chart->setSeries();
  725
  726   chart->setPlotArea();
  727
  728   chart->setGrid();
  729
  730   chart->setAxes();
  731
  732   chart->setLabels();
  733
  734   chart->setXLabels();
  735
  736   chart->setYLabels();
  737
  738   chart->setXAxis();
  739
  740   chart->setYAxis();
  741
  742   chart->setLegend();
  743
  744   chart->setSeries();
  745
  746   chart->setPlotArea();
  747
  748   chart->setGrid();
  749
  750   chart->setAxes();
  751
  752   chart->setLabels();
  753
  754   chart->setXLabels();
  755
  756   chart->setYLabels();
  757
  758   chart->setXAxis();
  759
  760   chart->setYAxis();
  761
  762   chart->setLegend();
  763
  764   chart->setSeries();
  765
  766   chart->setPlotArea();
  767
  768   chart->setGrid();
  769
  770   chart->setAxes();
  771
  772   chart->setLabels();
  773
  774   chart->setXLabels();
  775
  776   chart->setYLabels();
  777
  778   chart->setXAxis();
  779
  780   chart->setYAxis();
  781
  782   chart->setLegend();
  783
  784   chart->setSeries();
  785
  786   chart->setPlotArea();
  787
  788   chart->setGrid();
  789
  790   chart->setAxes();
  791
  792   chart->setLabels();
  793
  794   chart->setXLabels();
  795
  796   chart->setYLabels();
  797
  798   chart->setXAxis();
  799
  800   chart->setYAxis();
  801
  802   chart->setLegend();
  803
  804   chart->setSeries();
  805
  806   chart->setPlotArea();
  807
  808   chart->setGrid();
  809
  810   chart->setAxes();
  811
  812   chart->setLabels();
  813
  814   chart->setXLabels();
  815
  816   chart->setYLabels();
  817
  818   chart->setXAxis();
  819
  820   chart->setYAxis();
  821
  822   chart->setLegend();
  823
  824   chart->setSeries();
  825
  826   chart->setPlotArea();
  827
  828   chart->setGrid();
  829
  830   chart->setAxes();
  831
  832   chart->setLabels();
  833
  834   chart->setXLabels();
  835
  836   chart->setYLabels();
  837
  838   chart->setXAxis();
  839
  840   chart->setYAxis();
  841
  842   chart->setLegend();
  843
  844   chart->setSeries();
  845
  846   chart->setPlotArea();
  847
  848   chart->setGrid();
  849
  850   chart->setAxes();
  851
  852   chart->setLabels();
  853
  854   chart->setXLabels();
  855
  856   chart->setYLabels();
  857
  858   chart->setXAxis();
  859
  860   chart->setYAxis();
  861
  862   chart->setLegend();
  863
  864   chart->setSeries();
  865
  866   chart->setPlotArea();
  867
  868   chart->setGrid();
  869
  870   chart->setAxes();
  871
  872   chart->setLabels();
  873
  874   chart->setXLabels();
  875
  876   chart->setYLabels();
  877
  878   chart->setXAxis();
  879
  880   chart->setYAxis();
  881
  882   chart->setLegend();
  883
  884   chart->setSeries();
  885
  886   chart->setPlotArea();
  887
  888   chart->setGrid();
  889
  890   chart->setAxes();
  891
  892   chart->setLabels();
  893
  894   chart->setXLabels();
  895
  896   chart->setYLabels();
  897
  898   chart->setXAxis();
  899
  900   chart->setYAxis();
  901
  902   chart->setLegend();
  903
  904   chart->setSeries();
  905
  906   chart->setPlotArea();
  907
  908   chart->setGrid();
  909
  910   chart->setAxes();
  911
  912   chart->setLabels();
  913
  914   chart->setXLabels();
  915
  916   chart->setYLabels();
  917
  918   chart->setXAxis();
  919
  920   chart->setYAxis();
  921
  922   chart->setLegend();
  923
  924   chart->setSeries();
  925
  926   chart->setPlotArea();
  927
  928   chart->setGrid();
  929
  930   chart->setAxes();
  931
  932   chart->setLabels();
  933
  934   chart->setXLabels();
  935
  936   chart->setYLabels();
  937
  938   chart->setXAxis();
  939
  940   chart->setYAxis();
  941
  942   chart->setLegend();
  943
  944   chart->setSeries();
  945
  946   chart->setPlotArea();
  947
  948   chart->setGrid();
  949
  950   chart->setAxes();
  951
  952   chart->setLabels();
  953
  954   chart->setXLabels();
  955
  956   chart->setYLabels();
  957
  958   chart->setXAxis();
  959
  960   chart->setYAxis();
  961
  962   chart->setLegend();
  963
  964   chart->setSeries();
  965
  966   chart->setPlotArea();
  967
  968   chart->setGrid();
  969
  970   chart->setAxes();
  971
  972   chart->setLabels();
  973
  974   chart->setXLabels();
  975
  976   chart->setYLabels();
  977
  978   chart->setXAxis();
  979
  980   chart->setYAxis();
  981
  982   chart->setLegend();
  983
  984   chart->setSeries();
  985
  986   chart->setPlotArea();
  987
  988   chart->setGrid();
  989
  990   chart->setAxes();
  991
  992   chart->setLabels();
  993
  994   chart->setXLabels();
  995
  996   chart->setYLabels();
  997
  998   chart->setXAxis();
  999
  1000  chart->setYAxis();
  1001
  1002  chart->setLegend();
  1003
  1004  chart->setSeries();
  1005
  1006  chart->setPlotArea();
  1007
  1008  chart->setGrid();
  1009
  1010  chart->setAxes();
  1011
  1012  chart->setLabels();
  1013
  1014  chart->setXLabels();
  1015
  1016  chart->setYLabels();
  1017
  1018  chart->setXAxis();
  1019
  1020  chart->setYAxis();
  1021
  1022  chart->setLegend();
  1023
  1024  chart->setSeries();
  1025
  1026  chart->setPlotArea();
  1027
  1028  chart->setGrid();
  1029
  1030  chart->setAxes();
  1031
  1032  chart->setLabels();
  1033
  1034  chart->setXLabels();
  1035
  1036  chart->setYLabels();
  1037
  1038  chart->setXAxis();
  1039
  1040  chart->setYAxis();
  1041
  1042  chart->setLegend();
  1043
  1044  chart->setSeries();
  1045
  1046  chart->setPlotArea();
  1047
  1048  chart->setGrid();
  1049
  1050  chart->setAxes();
  1051
  1052  chart->setLabels();
  1053
  1054  chart->setXLabels();
  1055
  1056  chart->setYLabels();
  1057
  1058  chart->setXAxis();
  1059
  1060  chart->setYAxis();
  1061
  1062  chart->setLegend();
  1063
  1064  chart->setSeries();
  1065
  1066  chart->setPlotArea();
  1067
  1068  chart->setGrid();
  1069
  1070  chart->setAxes();
  1071
  1072  chart->setLabels();
  1073
  1074  chart->setXLabels();
  1075
  1076  chart->setYLabels();
  1077
  1078  chart->setXAxis();
  1079
  1080  chart->setYAxis();
  1081
  1082  chart->setLegend();
  1083
  1084  chart->setSeries();
  1085
  1086  chart->setPlotArea();
  1087
  1088  chart->setGrid();
  1089
  1090  chart->setAxes();
  1091
  1092  chart->setLabels();
  1093
  1094  chart->setXLabels();
  1095
  1096  chart->setYLabels();
  1097
  1098  chart->setXAxis();
  1099
  1100  chart->setYAxis();
  1101
  1102  chart->setLegend();
  1103
  1104  chart->setSeries();
  1105
  1106  chart->setPlotArea();
  1107
  1108  chart->setGrid();
  1109
  1110  chart->setAxes();
  1111
  1112  chart->setLabels();
  1113
  1114  chart->setXLabels();
  1115
  1116  chart->setYLabels();
  1117
  1118  chart->setXAxis();
  1119
  1120  chart->setYAxis();
  1121
  1122  chart->setLegend();
  1123
  1124  chart->setSeries();
  1125
  1126  chart->setPlotArea();
  1127
  1128  chart->setGrid();
  1129
  1130  chart->setAxes();
  1131
  1132  chart->setLabels();
  1133
  1134  chart->setXLabels();
  1135
  1136  chart->setYLabels();
  1137
  1138  chart->setXAxis();
  1139
  1140  chart->setYAxis();
  1141
  1142  chart->setLegend();
  1143
  1144  chart->setSeries();
  1145
  1146  chart->setPlotArea();
  1147
  1148  chart->setGrid();
  1149
  1150  chart->setAxes();
  1151
  1152  chart->setLabels();
  1153
  1154  chart->setXLabels();
  1155
  1156  chart->setYLabels();
  1157
  1158  chart->setXAxis();
  1159
  1160  chart->setYAxis();
  1161
  1162  chart->setLegend();
  1163
  1164  chart->setSeries();
  1165
  1166  chart->setPlotArea();
  1167
  1168  chart->setGrid();
  1169
  1170  chart->setAxes();
  1171
  1172  chart->setLabels();
  1173
  1174  chart->setXLabels();
  1175
  1176  chart->setYLabels();
  1177
  1178  chart->setXAxis();
  1179
  1180  chart->setYAxis();
  1181
  1182  chart->setLegend();
  1183
  1184  chart->setSeries();
  1185
  1186  chart->setPlotArea();
  1187
  1188  chart->setGrid();
  1189
  1190  chart->setAxes();
  1191
  1192  chart->setLabels();
  1193
  1194  chart->setXLabels();
  1195
  1196  chart->setYLabels();
  1197
  1198  chart->setXAxis();
  1199
  1200  chart->setYAxis();
  1201
  1202  chart->setLegend();
  1203
  1204  chart->setSeries();
  1205
  1206  chart->setPlotArea();
  1207
  1208  chart->setGrid();
  1209
  1210  chart->setAxes();
  1211
  1212  chart->setLabels();
  1213
  1214  chart->setXLabels();
  1215
  1216  chart->setYLabels();
  1217
  1218  chart->setXAxis();
  1219
  1220  chart->setYAxis();
  1221
  1222  chart->setLegend();
  1223
  1224  chart->setSeries();
  1225
  1226  chart->setPlotArea();
  1227
  1228  chart->setGrid();
  1229
  1230  chart->setAxes();
  1231
  1232  chart->setLabels();
  1233
  1234  chart->setXLabels();
  1235
  1236  chart->setYLabels();
  1237
  1238  chart->setXAxis();
  1239
  1240  chart->setYAxis();
  1241
  1242  chart->setLegend();
  1243
  1244  chart->setSeries();
  1245
  1246  chart->setPlotArea();
  1247
  1248  chart->setGrid();
  1249
  1250  chart->setAxes();
  1251
  1252  chart->setLabels();

```