Praca nad projektem TPCReco i jego konserwacja

Jakub Żak Jakub Korsak Tymon Maciejak

Opiekun: dr hab. Artur Kalinowski

1 Wstęp

Program *TPCReco* służy do analizy danych z detektora *ELITPC*[18], który bada reakcje fotonuklearne metodą monochromatycznych promieni gamma. Został on napisany w języku `C++` i jest aktywnie rozwijany na Wydziale Fizyki UW. Czerpie w dużej mierze z genewskiej biblioteki do analizy danych *ROOT*[1].

2 Cel projektu

Celem projektu było zaimplementowanie współpracy w programowaniu wspólnego projektu w języku `C++`. Zadaniem uczestników projektu było polepszenie ogólnej jakości kodu, poprawa wydajności oraz wprowadzenie metodologi *Test Driven Development*[2]. Przez korzystanie z systemu wersjonowania kodu (*GIT*[3]) oraz oprogramowania służącego do konteneryzacji projektu (*Docker*[4]), uczestnicy przyswoili najważniejesze informacje dotyczące praktycznej pracy nad kodem w zespole.

3 Kod projektu

Repozytorium projektu znajduje się pod adresem `https://github.com/akalinow/TPCReco`. Tam wprowadzano wszelkie zmiany za pomocą programu *GIT*. Każdy z uczestników tworzył własne gałęzie, na których pracował. Gałęzie utworzone przez uczestników to:

1. `develJZ`
2. `develZPS_tests`
3. `jakubkorsakdevel`
4. `README_update`.

Gałęzią najbardziej aktualną jest `develZPS_tests`[16]

4 Wprowadzone zmiany

Poniżej wymieniono główne zmiany w kodzie, przypisy są odnośnikami do odpowiednich zmian na serwisie *Github*.

- Refaktoryzacja [10] [11] [12]

  Zmieniono definicję liczby π,

  Zmieniono w odpowiednich miejscach typy z `int` na bardziej szczegółowe klasy enum `projection`, `direction`

  Uwspólniono sprawdzenie `DIR_U`, `DIR_V`, `DIR_W` na `IsDIR_UVW`

  Zamieniono użycie `NULL` na `nullptr`

  Zwiększono użycie kontenerów z biblioteki standardowej

  Zmiana zwykłych wskaźników na sprytne wskaźniki [5] [6]

  Przemianowano stałe z `#define` na `constexpr auto`

- Uogólniono użycie MultiKey [7]
- Stworzono wersję programu bez interfejsu graficznego [8]

  Dzięki temu, można było odpalać projekt na większym zakresie środowisk.

- Zaimplementowano *Singleton Design Pattern*

  Przełożono klasę `GeometryTPC` na singleton [9]

- Usprawnino zarządzanie pamięcią [13]

  Zaimplementowano elementy *Generic Programming* tworząc m.in. template `load_var` w `GeometryTPC`.

  Pozamieniano `std::map<...>` na `std::set<std::tuple<...>>` [14]

  Wszędzie gdzie można było, zamieniono zwykłe wskaźniki na sprytne.

- Ułatwiono zarządzanie plikami źródłowymi (danymi z eksperymentu). Teraz ścieżkę do danych można zmieniać niezależnie od kodu źródłowego, co na pewno ułatwi pracę z programem.

- Skonfigurowano projekt do współpracy z biblioteką *GoogleTest*[15]

  Projekt *GoogleTest* został wybrany ze względu na jego dojrzałość, przejrzystą dokumentację oraz szerokie wsparcie od społeczności.

- Przygotowano przykładowy test jednostkowy [16].

  W samym teście sprawdzono działanie funkcji `GeometryTPC::MatchCrossPoint`

  Opierając się na nim, będzie można pisać kolejne testy i zwiększając tym *Code Coverage* projektu.[17]

5 Podsumowanie

Projekt wykonano podczas semestru zimowego roku akademickiego 2020 na Wydziale Fizyki UW. W wyniku prac, poprawiono czytelność i spójnośc kodu, zoptymalizowano działanie programu oraz przygotowano podstawy do implementacji unit-testów. Dzięki temu, uczestnicy zrealizowali ogólne założenia projektu. W chwili zakończenia Projektu, kod nie działał poprawnie i wymaga dalszej pracy.

Literatura

[1] The ROOT Project, `https://root.cern.ch/`

[2] The Agile Alliance, `https://www.agilealliance.org/glossary/tdd/`

[3] GIT - The Stupid Content Manager, `https://git-scm.com/`

[4] Docker, `https://www.docker.com/`

[5] `https://github.com/akalinow/TPCReco/commit/4e0d7f74fd3dec64bab370e0b310717a8f45deb5`

[6] Microsoft Docs: Smart pointers - Modern C++, `https://docs.microsoft.com/en-us/cpp/cpp/smart-pointers-modern-cpp?view=vs-2019`

[7] `https://github.com/akalinow/TPCReco/commit/243bc8d7990572e5e475cdeebd228f120960e078`

[8] `https://github.com/akalinow/TPCReco/commit/36feb07949eb3e8e5884d024fe99f6cfe0b45d41`

[9] `https://github.com/akalinow/TPCReco/commit/ee33baa25274f396153d9d17349f5b664042389e`

[10] https://github.com/akalinow/TPCReco/commit/22cd90d8ec9e836789f6b1437aeeedb12f76fae0

[11] https://github.com/akalinow/TPCReco/commit/3802f4ed78487a68018c7bc7da6ae8b76090822d

[12] https://github.com/akalinow/TPCReco/commit/1458eeecd84c86c7f38d4639020addc92610967a

[13] https://github.com/akalinow/TPCReco/commit/640d12c0dbd14e04936410bc30ae40663064e2af

[14] https://github.com/akalinow/TPCReco/commit/4cdea1426ab5f8e4967c962e9bb4bbb025ddeff3

[15] Google test - biblioteka do testów, https://github.com/google/googletest

[16] https://github.com/akalinow/TPCReco/commit/a9defced06ec22a5ee7128ad96a3c49df58f2f65

[17] Wikipedia: Code Coverage, https://en.wikipedia.org/wiki/Code_coverage

[18] Prezentacja na temat ELITPC, https://drive.google.com/open?id=1ysWmcq72yF7J8-0beLsETGwes95He9IH