Projekt Supermarkt

Ziele

- Implementierung eines Supermarktes
- Integration einer GUI zur nutzerfreundlichen Bedienung
- Datenerfassung und -speicherung in seperaten Dateien bzw. Ordnern

Hauptfunktionen

- Verwaltung von Kunden, Händlern, Produkten und Warenkörben
- Export von Kassenzetteln, Inventur- und Kundendateien
- Trennung von Datenzugriff (`ReadData`) und Auswertung (`Statistik`)
- Multithreading-fähiger Dateizugriff

CMakeLists.txt

- Zentrales Dokument zur Steuerung vom Compile-Vorgang
- **src:** Ablageort für die Source-Dateien
- **inc:** Ablageort für die Header-Dateien, Unterteilung in base und utils
- **Verwendeter Standard:** CXX 17
- **ccache:** Zwischenspeichern der Build-Objekte im Cache zur Beschleunigung vom Build-Vorgang
- **Optimierung O2:** Beschleunigung vom Build-Vorgang und Debugging sind möglich

.vscode

- **tasks.json:** JSON-Datei fürs Steuern vom Kompilier-Vorgang inkl. Shortcuts für die Tastatur
- **settings.json:** JSON-Datei zur Konfiguration für C++

build

- Speicherort fuer die Executables, Binärdateien und Befehle fürs Komplieren

inc

- Ablageort für die Header-Dateien
- **base:** Unterordner für die einfachen Klassen
- **utils:** Unterordner für die Template-Klassen

src

- Speicherort für die Source-Dateien
- **tests.cpp:** Test-File fuer alle Executables

Integration von C++ in Python

Erweiterungen in der CMakeLists.txt:**

- **Pybind11 einbinden**

  add_subdirectory(pybind11) find_package(Python3 COMPONENTS Interpreter Development REQUIRED)

- **pybind11_bindings mit Python-Bibliotheken linken**

  target_link_libraries(supermarkt PRIVATE Python3::Python)
  include_directories(${Python3_INCLUDE_DIRS})
  include_directories(${PROJECT_SOURCE_DIR}/include)
  include_directories(${CMAKE_SOURCE_DIR}/pybind11/include)

- **Modul für Python (Statistik-Pybind11)**

  pybind11_add_module(py_bindings Verwenden der gleichen Executables aus src, aber ohne main.cpp)

- **Linken von py_bindings mit poppler-cpp fuer Konventierung von PDFs in String**

  target_link_libraries(py_bindings PRIVATE poppler-cpp)

Erweiterungen fuer settings.json:

```
"python.envFile": "${workspaceFolder}/.env",
"python.analysis.extraPaths": [
"./build"
],
"python.analysis.stubPath": "./build",
"python.analysis.autoSearchPaths": true,
"python.analysis.diagnosticSeverityOverrides": {
    "reportMissingImports": "none",
    "reportMissingModuleSource": "none"
}
```

Beheben von Pylint-Meldung, "Module couldn't be found":

- **Inhalt von .pylintrc:**

  [MASTER] init-hook='import sys; sys.path.append("build")'

- **Inhalt von.env:** PYTHONPATH=./build

- **Temporäre Eingabe via Terminal:**

  export PYTHONPATH=/home/arrif-sondjalim/c++projects/supermarkt/build

Zu erledigende Aufgaben

- GUI: Implementierung in Python