Pokročilé programování v jazyce C pro chemiky (C3220) – **závěrečné cvičení**

1. Program v Qt zobrazující sekvenci proteinu z PDB souboru

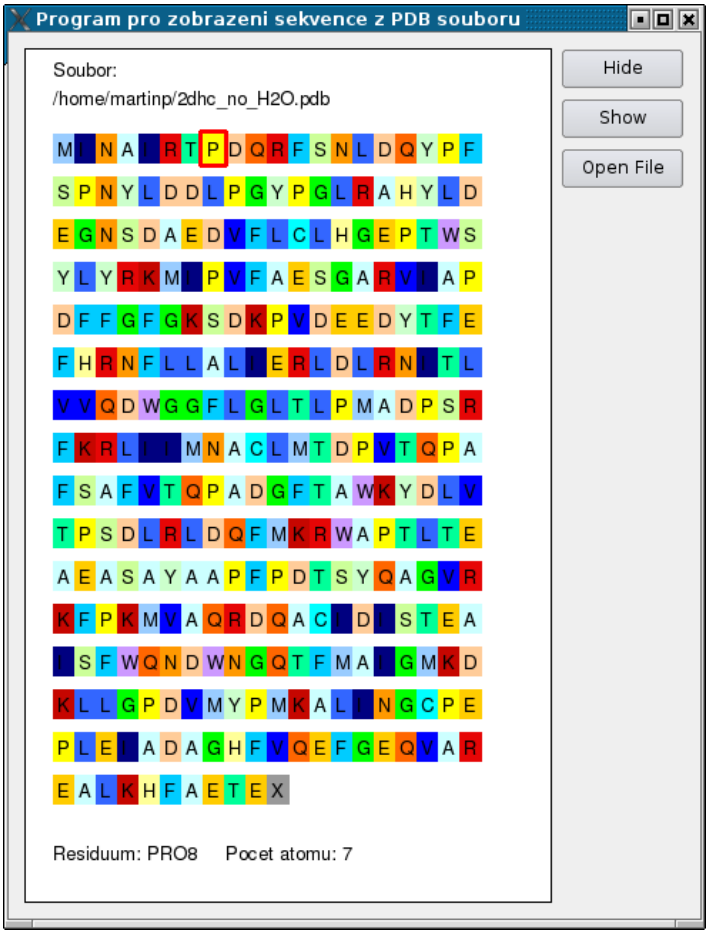
Zadání

Vytvořte program s pomocí knihovny Qt, který bude schopen načíst PDB soubor s proteinem a zobrazit sekvenci residuí.

Program bude mít následující vlastnosti:

- Program bude obsahovat tlačítko "Open File" po jehož stisknutí se otevře dialogové okno pro výběr PDB souboru. Potom se PDB soubor načte a zobrazí se sekvence proteinu.
- Sekvence bude zobrazena tak, že pro každé residuum se zobrazí malý barevný obdélník (použijte barvy uvedené níže). Uprostřed obdélníku bude jednopísmenná zkratka residua.
- V horní části hlavního widgetu bude zobrazen název PDB souboru.
- Program bude dále opatřen dvěma tlačítky "Hide" a "Show", která umožní skrýt a opět zobrazit jednopísmenné zkratky residuí (tj. při skrytí se bude pro každé residuum zobrazovat jen barevný obdélník bez jednopísmenné zkratky)
- Pokud uživatel klikne myší na obdélník residua, označí se toto residuum tak, že se kolem něj nakreslí červený rámeček. Navíc se v dolní části zobrazí informace o residuu: třípísmenná zkratka, číslo residua a počet atomů residua.

Program otestujte se strukturou crambinu (*1jxy_noal.pdb*) a enzymu haloalkan dehalogenáza (*2dhc_no_H2O.pdb*). Oba soubory naleznete v IS MU ve složce *data*.



Dodržujte následující pravidla

- Na začátek každého hlavičkového souboru vždy uveďte direktivy preprocesoru:
  ```
  #ifndef FILENAME_H
  #define FILENAME_H
  ```
  a na konci souboru:
  ```
  #endif
  ```
- Pro jména argumentů metod a lokální proměnné nikdy nepoužívejte stejná jména, která jste již použili pro členské proměnné dané třídy. Došlo by k překrytí těchto členských proměnných, což je matoucí.
- Nezapomeňte inicializovat všechny členské proměnné třídy. Toto se týká jen proměnných základních typů, proměnné objektových typů jsou inicializovány svými vlastními konstruktory. Inicializaci provádějte pokud možno přímo v definici položek v těle třídy, případně v inicializačním seznamu konstruktoru třídy. Přiřazení v těle konstruktoru používejte jen v dobře odůvodněných případech.
- Datové členy třídy uvádějte vždy jak soukromé. V metodách třídy přistupujte k těmto členským proměnným přímo, pro přístup odjinud použijte příslušné přístupové metody (`getXXX()`, `setXXX()`).
- Metody, které nemění data třídy, definujte jako konstantní.
- Používejte přednostně standardní kontejnery (např. *vector*) místo běžných polí a obyčejných ukazatelů.
- Ke správě dynamicky alokovaných dat použijte chytré ukazatele či hierarchii QObject (ukazatele na rodičovský objekt předávané konstruktoru). Ruční správu paměti pomocí kombinace operátorů `new` a `delete` používejte jen v případě nutnosti.
- V případě potřeby použijte dokumentaci ke knihovně Qt: https://doc.qt.io/qt-5/classes.html
- Přidáte-li do souboru **.cpp* nebo **.h* hlavičkový soubor, je potřeba znovu vygenerovat *Makefile* (příkazem *qmake *.pro*). Toto se netýká knihovních hlavičkových souborů.

Nápověda

1. Za základ použijte program z úlohy 2 ze cvičení 10.
2. Do souboru *atom.h* zkopírujte definici třídy `Atom` z úlohy 1 ze cvičení 8. Do souboru *atom.cpp* zkopírujte metody třídy `Atom`. Kromě metod `readLine()` a `writeLine()` musí třída `Atom` obsahovat také metody pro přístup k datovým členům `residueNumber` a `residueName` (tj. metody `getResidueNumber()`, ...).
3. V souboru *residue.h* definujte třídu `Residue`, která bude obsahovat datové členy `atomFirst`, `atomLast` (pořadí prvního a posledního atomu daného residua v kontejneru atomů), `residueName`, `residueNumber` (jméno a číslo residua, jak je uvedeno v PDB souboru). Třída bude obsahovat konstruktor `Residue(int atomFirst, const string &residueName, int residueNumber)`, dle potřeby i konstruktor bez parametrů. Třída bude dále obsahovat metody pro přístup ke členům třídy (`getAtomLast()`, `setAtomLast()`, ...). V souboru *residue.cpp* implementujte příslušné metody třídy `Residue`.
4. Vytvořte soubor projektu (*qmake -project*) a *Makefile* (*qmake *.pro*). Přeložte a odstraňte chyby.

5. Do třídy `GraphicWidget` přidejte kontejner pro seznam atomů (`vector<Atom> atoms;`) a residuí (`vector<Residue> residues;`). Nezapomeňte vložit potřebné hlavičkové soubory do souboru *graphicwidget.h*.
6. Do třídy `GraphicWidget` přidejte metodu `readPdbFile()` která načte řádky ATOM a HETATM z PDB souboru, jehož jméno jí bude předáno jako argument. Metoda bude stejná jako v úloze 1 ze cvičení 8.
7. Z metody `GraphicWidget::openFile()` zavoláme metodu `readPdbFile()` a jako argument jí předáme jméno souboru, které bylo vybráno uživatelem v dialogovém okně. Poté si necháme vypsat kontrolní výpis atomů (tj. pro každý atom v kontejneru `atoms` voláme metodu `writeLine()`). Otestujeme správnost načítání PDB souboru.
8. Ve třídě `GraphicWidget` implementujte metodu `findResidues()`, která naplní kontejner `residues`. Metoda bude procházet kontejner atomů a bude vyhledávat residua. Pro každé residuum přidané do kontejneru nastaví jméno a číslo residua a také index prvního a posledního atomu v kontejneru `atoms`. Metoda bude pracovat podobně jako obdobná funkce používaná v minulém semestru (úloha 9.1). Na konec metody `findResidues()` přidejte testovací výpis, který vypíše všechna residua v kontejneru `residues`.
9. Metoda `findResidues()` bude volána z `GraphicWidget::openFile()` ihned po zavolání `readPdbFile()`. Program otestujte a zkontrolujte správnost vypsaných residuí.
10. Do třídy Residue přidejte metodu `getResidueChar()`, která vrátí znak (typu `char`) odpovídající typu residua (viz žlutý rámeček dole vlevo). Pro neznámé residuum vraťte znak X. Pro překlad názvu residua na znak lze jednoduše použít 20 podmínek, i když elegantnějším řešením je přidat do třídy `Residue` privátní položku `static const unordered_map<string, char> nameToChar`. Tu je pak třeba v souboru *residue.cpp* inicializovat (na globální úrovni, mimo funkce) stylem `const unordered_map<string, char> Residue::nameToChar = {{"ALA", 'A'}, {"ARG", 'R'}}` atd.. Pro samotný překlad pak stačí volat na mapě metodu `find()`, která vrací iterátor na nalezený záznam. Pokud hledání neuspěje, iterátor je roven `end()`.
11. Do metody `GraphicWidget::paintEvent()` implementujte vykreslování residuí. Je možné vykreslovat např. 20 residuí na řádek. Prozatím se budou vypisovat jen znaky, jeden pro každé residuum. Pro předání znaku metodě `QPainter::drawText()` použijte `QString(QChar(znak))`
12. Do třídy `Residue` přidejte metodu `void getColorRgb(int &r, int &g, int &b)`, která vrátí 3 složky barvy odpovídající typu residua (viz žlutý rámeček dole vpravo). Pro neznámé residuum vrátí šedou barvu (153 153 153). Tuto metodu využijte v `GraphicWidget::paintEvent()` pro nakreslení barevného obdélníčku residua.
13. Nyní je třeba implementovat reakci na kliknutí myší na obdélník residua. Pro tento účel přidáme do třídy `Residue` proměnné `posX` a `posY` a metody pro manipulaci s nimi `setPosXY()`, `getPosX()`, `getPosY()`. Do těchto proměnných vždy uložíme hodnoty souřadnice levého horního rohu obdélníčku residua, tyto hodnoty nastavíme v metodě `GraphicWidget::paintEvent()` vždy ihned po nakreslení obdélníčku.
14. Do třídy `GraphicWidget` přidejte ukazatel `Residue* selectedResidue`, který bude ukazovat na objekt aktuálně označeného residua. Do metody `GraphicWidget::mousePressEvent()` implementujte kód, který projde všechna residua v `residues` a bude testovat, zdali se myš nachází v obdélníčku residua. Pokud ano, nastaví na toto residuum ukazatel `selectedResidue`. Nakonec požádá o překreslení okna. Do `GraphicWidget::paintEvent()` je třeba přidat kód, který nakreslí červený rámeček kolem residua odkazovaného `selectedResidue`.
15. Zobrazení názvu souboru v okně bude realizováno podobně jako v úloze 2 ze cvičení 10, pouze jméno bude zobrazeno v horní části okna.
16. Pro vypsání počtu atomů residua přidáme do třídy `Residues` metodu `getAtomCount()`, která vrátí počet atomů residua spočítaný z rozdílu `atomLast` a `atomFirst`. Následující kód ukazuje, jak zobrazíme název residua a počet atomů v metodě `GraphicWidget::paintEvent()`. Nezapomeňte vložit hlavičkový soubor *sstream*.

```
ostringstream ss;
ss << "Residuum: ";
ss << selectedResidue->getResidueName();
ss << selectedResidue->getResidueNumber();
ss << "     Pocet atomu: ";
ss << selectedResidue->getAtomCount();

painter.drawText(20, height() - 30,
      ss.str().c_str());
```

```
ALA  A
ARG  R
ASN  N
ASP  D
CYS  C
GLN  Q
GLU  E
GLY  G
HIS  H
ILE  I
LEU  L
LYS  K
MET  M
PHE  F
PRO  P
SER  S
THR  T
TRP  W
TYR  Y
VAL  V
```

```
ALA  204  255  255
ARG  230    6    6
ASN  255  153    0
ASP  255  204  153
CYS    0  255  255
GLN  255  102    0
GLU  255  204    0
GLY    0  255    0
HIS  255  255  153
ILE    0    0  128
LEU   51  102  255
LYS  198    6    0
MET  153  204  255
PHE    0  204  255
PRO  255  255    0
SER  204  255  153
THR    0  255  153
TRP  204  153  255
TYR  204  255  204
VAL    0    0  255
```