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

1. Program v Qt zobrazující sekvenci proteinu z PDB souboru

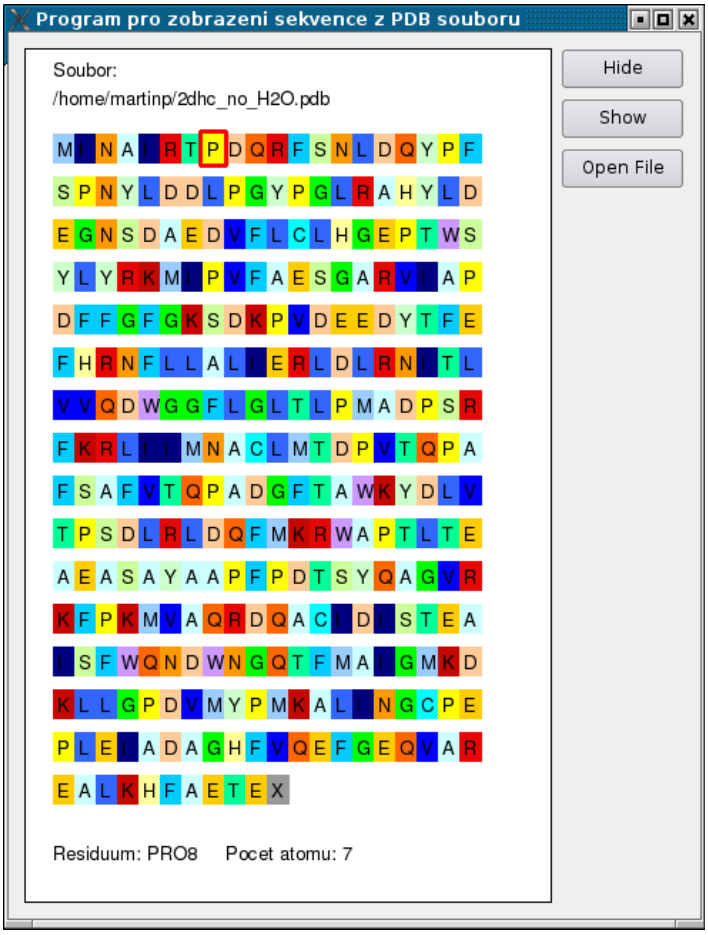
Zadání

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 halolkan dehalogenáza (*2dhc_no_H2O.pdb*).



Dodržujte následující pravidla

- Na začátek každého hlavičkového souboru vždy uveďte direktvy předprocesoru:
  ```
  #ifndef FILENAME_H
  #define FILENAME_H
  ```
  a na konci souboru:
  ```
  #endif
  ```
- Pro jména argumentů metod a lokální proměnné nikdy nepoužívejte stejná jména jako jste již použili pro členské proměnné dané třídy. Došlo by k překrytí těchto členských proměných a program by se choval jinak než očekáváte.
- Nezapomeňte v konstruktoru inicializovat všechny členské proměnné třídy. Toto se týká jen proměnných základních typů, proměnné objektových typů se inicializují ve svých vlastních konstruktorech. Řetězce typu `string` je však také vhodné inicializovat prázdným řetězcem;
- Datové členy třídy uvádějte vždy jak soukromé. V metodách třídy přistupujte k těmto členským proměnným přímo, pouze v případě že k nim potřebujete přistupovat od jinud použijte příslušné přístupové metody (`getXXX()`, `setXXX()`).
- Metody, které nemění data definujte jako konstantní.
- Nezapomeňte uvolnit všechna dynamicky alokovaná data (tj. alokovaná pomocí operátoru `new`). Data zpravidla uvolňujeme v destruktorech.
- V případě potřeby použijte dokumentaci ke knihovně Qt: http://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 týká pouze situace kdy přidáváte vámi vytvořené hlavičkové soubory nikoli knihovní hl. soubory.

Nápověda

1. Za základ použijte program z úlohy 2 ze cvičení 10.
2. Vytvořte soubory *atom.cpp*, *atom.h*, *residue.cpp*, *residue.h*. Do hlavičkových souborů nezapomeňte uvést příslušné direktivy předprocesoru (`#ifndef` ...).
3. Do souboru *atom.h* zkopírujte definici třídy `Atom` z úlohy 1 ze cvičení 8. Na začátek je třeba vložit hlavičkové soubory *iostream* a *string* a dekladovat `using namespace std;` Do souboru *atom.cpp* zkopírujte metody třídy `Atom`, na začátek vložte potřebné hlavičkové soubory (*fstream*, *sstream*, *iomanip*, *atom.h*). Kromě metod `readLine()`, `writeLine()` a `print()` musí třída `Atom` obsahovat také metody pro přístup k datovým členům `residueNumber` a `residueName` (tj. metody `getResidueNumber()`, ...).
4. 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 bez argumentů a také je užitečné přidat konstruktor `Residue(int argAtomFirst, const string &argResidueName, int argResidueNumber)`. Třída bude dále obsahovat metody pro přístup ke členům třídy (`getAtomFirst()`, `setAtomFirst()`, ...). V souboru *residue.cpp* implementujte příslušné metody třídy `Residue`.

5. Vytvořte soubor projektu (*qmake -project*) a *Makefile* (*qmake *.pro*). Přeložte a odstraňte chyby.
6. Do třídy `GraphicWidget` přidejte kontejner pro seznam atomů (`vector<Atom*> atomsContainer;`) a residuí (`vector<Residue*> residuesContainer;`). Nezapomeňte vložit hlavičkové soubory (*vector*, *atom.h*, *residue.h*) do souboru *residue.h*.
7. Po každém přidání hlavičkových souborů do některého ze zdrojových souborů (**.cpp*, **.h*) je potřeba znovu vygenerovat *Makefile* (*qmake *.pro*) a přeložit program.
8. 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. Do souboru *graphicwidget.cpp* musíme přidat hlavičk. soubor *fstream*.
9. 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 `atomsContainer` voláme metodu `print()`). Otestujeme správnost načítání PDB souboru.
10. Ve třídě `GraphicWidget` implementujte metodu `setResidues()`, která naplní kontejner `residuesContainer`. Metoda bude procházet kontejner atomů a bude vyhledávat rsidua. Pro každé residuum přidané do kontejneru nastaví jméno a číslo residua a také index prvního a posledního atomu (indexem se zde rozumí pořadí atomu v kontejneru `atomsContainer`). Metoda bude pracovat podobně jako obdobná funkce používaná v minulém semestru (úloha 9.1). Na konec metody `setResidues()` přidejte testovací výpis, který vypíše všechna residua v kontejneru `residuesContainer`.
11. Metoda `setResidues()` bude volána z `GraphicWidget::openFile()` ihned po zavolání `readPdbFile()`. Program otestujte a zkontrolujte správnost vypsaných residuí.
12. 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. Pozn.: kód lze zapsat jednoduše formou 20 podmínek, i když existují i efektivnější řešení.
13. 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. Protože metoda `QPainter::drawText()`, přijímá jako třetí argument hodnotu typu `QString`, musíme jí předat znak residua následovně : `painter.drawText(x, y, QString( QChar(residuesContainer[i]->getResidueChar())));`
14. Do třídy `Residue` přidejte metodu `void getColorRgb(int &r, int &g, int &b)`, která vrátí 3 složky barev odpovídající typu residua (viz. žlutý rámeček dole vpravo). Pro neznámé residuum vrátí šedou barvu (153 153 153). Tuto metodu využijeme v `GraphicWidget::paintEvent()` pro nakreslení barevného čtverečku residua.
15. 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 čtverečku residua, tyto hodnoty nastavíme v metodě `GraphicWidget::paintEvent()` vždy ihned po nakreslení čtverečku.
16. Do třídy `GraphicWidget` přidejte proměnnou `selectedResidue` do které vždy přiřadíme index (tj. pořadí v kontejneru `residuesContainer`) aktuálně označeného residua (tj. residua na jehož čtvereček uživatel kliknul myší). Do metody `GraphicWidget::mousePressEvent()` implementujte kód, který projde všechna residua v `residuesContainer` a bude testovat, zda-li se myš nachází ve čtverečku residua. Pokud ano, přiřadí index tohoto residua do `selectedResidue`. Nakonec požádá o překreslení okna. Do `GraphicWidget::paintEvent()` je třeba přidat kód. který nakreslí červený rámeček kolem residua jehož index je `selectedResidue`.
17. Zobrazení názvu souboru v okně bude realizováno podobně jako v úloze 2 ze cvičení 10, pouze jméno bude zobrazeno v horní části okna.
18. Pro vypsání počtu atomů residua přidáme do třídy `Residues` metodu `getAtomsCount()`, která vrátí počet atomů residua (spočítá ho jako rozdíl`atomLast–atomFirst`). Následující kód ukazuje, jak zobrazíme název residua a počet atomů v metodě `GraphicWidget::paintEvent()`. Nezapomeňte vložit hlavičkový soubor *sstream*.

```
string residueDescription;
ostringstream ss;
ss << "Residuum: ";
ss << residuesContainer[selectedResidue]
        ->getResidueName();
ss << residuesContainer[selectedResidue]
        ->getResidueNumber();
ss << "     Pocet atomu: ";
ss << residuesContainer[selectedResidue]
        ->getAtomsCount();
residueDescription = ss.str();

if (residueDescription.length() > 0)
  painter.drawText(20, height() - 30,
        residueDescription.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