Praktikum iz objektno orijentisanog programiranja (13S112POOP)
Projektni zadatak – C++

Napisati skup klasa sa odgovarajućim metodama, konstruktorima, operatorima i destruktorima za realizaciju softverskog sistema za poslovanje na berzi. Potrebno je obezbediti prikupljanje podataka o istoriji cena akcija, izdvajanje (parsiranje) i analizu prikupljenih podataka, kao i sistem za simulaciju kupovine i prodaje akcija. O elementima berzanskog poslovanja se više informacija može videti na https://skolaberze.net/recnik-pojmova/.

Korisnik (naručilac) softvera, želi da softver pruži sledeće funkcionalnosti:

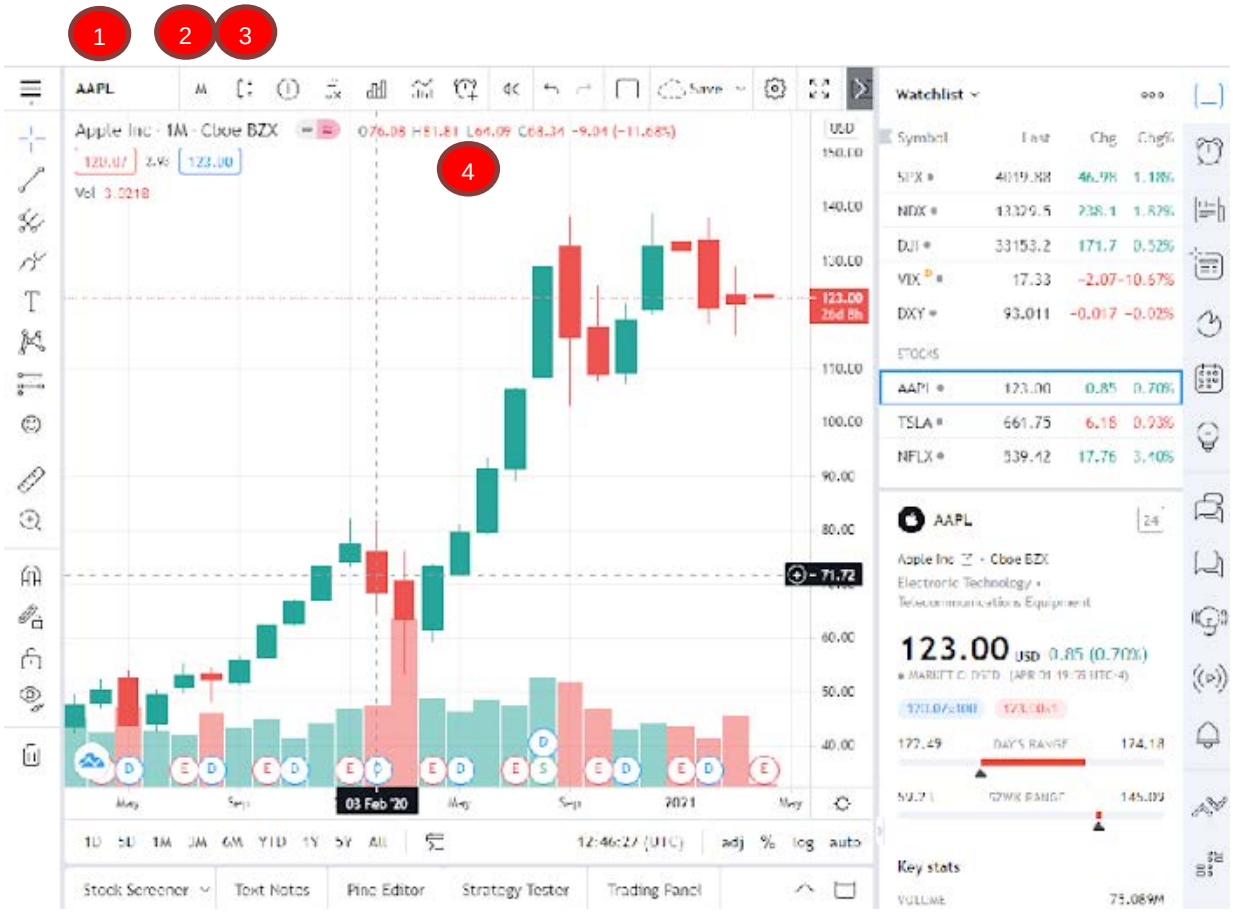
- Interakciju sa korisnikom putem tekstualnog menija (konzole) ili grafičkog korisničkog interfejsa
- Prikupljanje stvarnih (*live*) podataka o cenama akcija
- Parsiranje prikupljenih podataka
- Osnovnu manipulaciju
    - Rad sa svećama (*candlestick*)
    - Izračunavanje statističkih pokazatelja (indikatora)
- Simulaciju kupovine i prodaje akcija
    - Otvaranje naloga za simulaciju
    - Kupovina i prodaja akcija
    - Portfolio korisnika – praćenje i analiza
- Kraj rada

Za uspešno rešenje zadatka potrebno je izvršiti analizu zahteva. Kao rezultat analize, potrebno je dopuniti i precizirati funkcionalnu specifikaciju softverskog alata. Na osnovu specifikacije, potrebno je napisati sistem klasa u jeziku C++ koje realizuju traženi softver. U nastavku su navedeni neki elementi specifikacije. Od studenata se očekuje da dopune one stavke koje nisu dovoljno precizno formulisane, odnosno dodaju nove stavke (tamo gde to ima smisla) ukoliko uoče prostor za unapređenje. Izmene i dopune specifikacije mogu da donekle odudaraju od zahteva naručioca softvera u onoj meri u kojoj to neće narušiti traženu funkcionalnost. Takođe, priloženi UML dijagram koji opisuje zahtevani softver se ne mora obavezno poštovati, već samo predstavlja skicu potencijalnog rešenja. Prilikom izrade specifikacije voditi računa o potencijalnom unapređenju softvera na osnovu naknadnih zahteva.

Prilikom izrade rešenja, od studenata se očekuje intenzivno korišćenje svih onih mogućnosti koje pružaju specifikacija jezika C++ i biblioteka STL, kao što su šablonske funkcije, kolekcije, algoritmi, regularni izrazi, iteratori, lambda izrazi i sl. **Rešenja koja ne vode računa o ovom aspektu neće moći da dobiju maksimalan broj poena.** Takođe, voditi računa o **objektno orijentisanom dizajnu rešenja**, čistoći, čitkosti i komentarisanju programskog koda.

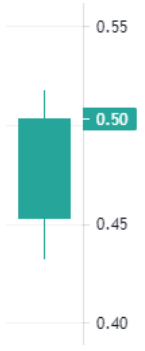
Osnovni pojmovi

U ovoj sekciji pojašnjeni su osnovni pojmovi u vezi sa berzanskim poslovanjem i praćenjem istorije cena akcije. Na slici 1 dat je primer alata za nadgledanje akcija (tradingview.com/chart/). U jednom trenutku moguće je posmatrati istoriju kretanja cene samo jedne akcije. Brojem 1 označen je simbol akcije koja se nadlgeda (AAPL je simbol koji koristi kompanija Apple). Brojem 3 označen je izbor prikaza istorije cena akcije pomoću sveća (crveni i zeleni stubići o kojima će više reči biti kasnije). Brojem 2 označen je izbor perioda na koji se odnosi jedna sveća (M – jedna sveća se odnosi na mesec dana). Prelaskom mišem preko neke od sveća dobijaju se informacije označene brojem 4. Na slici 2 izdvojena je i objašnjena jedna sveća.



Slika 1 Primer prikaza istorije cena akcije pomoću sveća

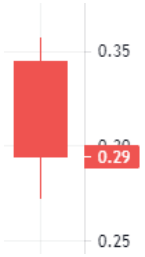
Jedna sveća prikazuje kretanje cene akcije u okviru izabranog vremenskog perioda. Na slici 1 prikazane su sveće koje oslikavaju period od jednog meseca, ali je moguće izabrati i druge intervale (1s, 1 min, 1h, 1 dan, 1 nedelja, 1 mesec itd.). Sveća može biti zelene ili crvene boje. Ukoliko je cena akcije u toku izabranog intervala porasla sveća će biti zelena, dok će u suprotnom biti crvena. Stub sveće (ispunjeni pravougaonik) određuje cenu na početku i na kraju intervala. Ukoliko se radi o zelenoj sveći, donja vrednost označava cenu na početku, a gornja na kraju intervala, dok je kod crvene sveće suprotno. Ekstremne vrednosti (minimum i maksimum) cene akcije u posmatranom intervalu su predstavljene vertikalnom linijom koja seče stub sveće. Na slici 2 cena akcije na početku intervala je bila $0.45.



Slika 2 Primer sveće

Na kraju intervala cena je bila $0.50, te je sveća zbog toga zelene boje. Maksimalna cena koju je akcija dostigla u posmatranom intervalu je $0.52, dok je minimalna $0.43.

Na slici 3 prikazan je još jedan primer sveće. Cena akcije na početku intervala je bila $0.34, dok je cena na kraju intervala $0.29, te je sveća stoga crvena. Najviša cena koju je akcija dostigla u posmatranom period je $0.36, a najniža $0.27.



Slika 3 Primer sveće II

Na slici 1 brojem 4 označene su četiri vrednosti koje opisuju jednu sveću:

- O (Open) – cena na početku intervala
- H (High) – najviša cena u toku intervala
- L (Low) – najniža cena u toku intervala
- C (Close) – cena na kraju intervala

Sveće koje će se koristit u ovom projektu odnose se na vremenski period od 1h.

Funkcionalna specifikacija

U nastavku je zadat deo korisničkih zahteva koje treba razraditi i, po potrebi, dopuniti tako da se dobije funkcionalna aplikacija.

Interakcija sa korisnikom

Korisnik može da interaguje sa programom bilo izborom odgovarajućih opcija iz tekstualnog menija putem tastature ili izborom u datom trenutku dostupnih opcija putem grafičkog korisničkog interfejsa. Nije potrebno realizovati oba načina. Interakcija u slučaju grafičkog interfejsa može da se vrši putem tastature ili miša. U zavisnosti od izabrane opcije i njenih parametara, program izvršava zadatu opciju ili ispisuje poruku greške. Poruka greške treba da bude što je moguće detaljnija da bi korisniku pomogla da grešku otkloni. Sve eventualne parametre koji su potrebni prilikom rada aplikacije je potrebno zatražiti od korisnika. Ukoliko korisnik ne zada ništa, koristiti vrednosti fiksirane u programu.

Registracija i prijavljivanje

Prilikom pokretanja programa korisnik se mora prijaviti koristeći korisničko ime i lozinku. Podatke o korisnicima je moguće nezaštićene čuvati u datoteci koja se nalazi na predefinisanoj lokaciji na računaru. Ukoliko korisnik nema nalog, može napraviti novi. Prilikom pravljenja naloga unosi se korisničko ime, lozinka i iznos novca sa kojim se započinje poslovanje na berzi. Podatke o stanju računa korisnika, kao i transakcijama koje obavlja (o čemu će više biti reči u nastavku) treba čuvati u posebnom direktorijumu, u zasebnoj datoteci za svakog korisnika.

Po prijavljivanju, korisniku se prikazuje glavni meni iz koga je dalje moguće koristiti u nastavku opisane funkcionalnosti:

- Prikupljanje podataka o određenoj akciji
- Prikaz podataka o određenoj akciji
- Računanje statističkih pokazatelja (indikatora) za određenu akciju
- Kupovina i prodaja akcija
- Prikaz portfolija korisnika
- Kraj rada

Prikupljanje podataka

Korisnik može zatražiti prikupljanje podataka o istoriji cene neke akcije, pri čemu zadaje simbol akcije i period za koji želi da dohvati podatke. Period se zadaje pomoću *timestamp* (broj sekundi od 1.1.1970) vrednosti početka i kraja. Podaci se prikupljaju u realnom vremenu (uživo) sa ***finance.yahoo.com*** sajta pomoću ***curl*** biblioteke. Rad sa ovom bibliotekom je opisan u dodatku postavke projekta. Prilikom upotrebe biblioteke potrebno je formirati URL zahteva. Primer URL-a je dat u nastavku:

*https://query1.finance.yahoo.com/v8/finance/chart/**aapl**?period1=**1617272670**&period2=**1617531870**&interval=1h*

Crveni deo teksta treba zameniti simbolom akcije za koju se traže podaci, a narandžasti deo periodom. Zadatak *curl* biblioteke je da dovuče tražene podatke u tekstualnom formatu.

Parsiranje podataka

Prikupljene podatke je potrebno parsirati, odnosno iz dohvaćenog teksta izvući korisne informacije. Podaci će biti dohvaćeni u JSON formatu koji se veoma često koristi za razmenu podataka, a koji je opisan u dodatku postavke projekta. Među prikupljenim podacima nalaze se informacije o svećama za interval od 1h (*Timestamp, Open*, *Close*, *High* i *Low*) za izabrani period. **Nije dozvoljena upotreba biblioteka za parsiranje JSON sadržaja!**

Prikaz podataka o određenoj akciji

Korisnik može zahtevati da mu se prikažu sveće akcije čiji simbol zada. Ukoliko podaci o akciji nisu dostupni, potrebno ih je prikupiti na opisani način. Prilikom prikaza informacija, u pojedinačnim redovima se tabelarno prikazuju informacije o svećama (*Timestamp*, *Open*, *Close*, *High* i *Low*) i to crvenom ili zelenom bojom u zavisnosti od boje sveće. Takođe, prilikom zadavanja simbola akcije čije podatke želi da vidi, korisniku treba ponuditi i opciju da se uz osnovne informacije o svećama ispisuju i statistički pokazatelji *Moving Average - MA(n)* i *Exponential Moving Average EMA(n)*. Parametar n unosi korisnik.

MA(n) predstavlja aritmetičku sredinu cene akcije (*Close* cena se uzima prilikom računanja) u poslednjih *n* intervala koji se posmatraju (u našem slučaju u poslednjih *n* sati). Drugim rečima, za svaku sveću prikazuje se aritmetička sredina prethodnih *n* sveća. Pomoću ovog indikatora prati se trend kretanja cene akcije.

EMA(n) je sličan *MA(n)* s tim što se sati ne razmatraju ravnopravno, već značaj opada kako se odlazi u prošlost. Računa se po sledećoj rekurzivnoj formuli:

$$EMA(n)_n = Close_n * \frac{2}{n+1} + EMA(n)_{n-1} * (1 - \frac{2}{n+1})$$

$$EMA(n)_1 = Close_1$$

Prikaz portfolija korisnika

Korisniku se prikazuje trenutno stanje na njegovom računu i spisak svih akcija koje trenutno poseduje. Za svaku kupovinu se tabelarno ispisuje ID kupovine, simbol akcije, broj kupljenih akcija, cena po kojoj je akcija kupljena, trenutna cena, razlika između trenutne cene i cene kupovine (apsolutna i relativna izražena u procentima). Razlika se ispisuje zelenim ili crvenim slovima u zavisnosti od toga da li je cena akcije od kako ju je korisnik kupio porasla ili opala.

Korisnik može zahtevati prodaju akcija koje poseduje, pri čemu navodi ID kupovine kojom su akcije kupljene i broj akcija koje želi da proda. Stanje na računu korisnika se uvećava za vrednost prodatih akcija (*broj_akcija* * *trenutna_cena_akcije*). Trenutna cena akcije se pri svakoj prodaji prikuplja na već opisani način sa sajta *finance.yahoo.com*.

Korisnik može zahtevati kupovinu akcija, pri čemu navodi simbol akcije koju želi da kupi i broj akcija. Kupovina se može ostvariti samo ukoliko korisnik ima dovoljno sredstava na računu. Kupovina se uvek obavlja po trenutnoj ceni akcije, koju je, kao i kod prodaje, potrebno prikupiti u realnom vremenu. Svaka kupovina ima svoj identifikator, koji se može formirati na proizvoljan način, ali mora biti jedinstven.

Kraj rada

Korisnik može da zahteva kraj rada. Od korisnika se traži potvrda za napuštanje programa.

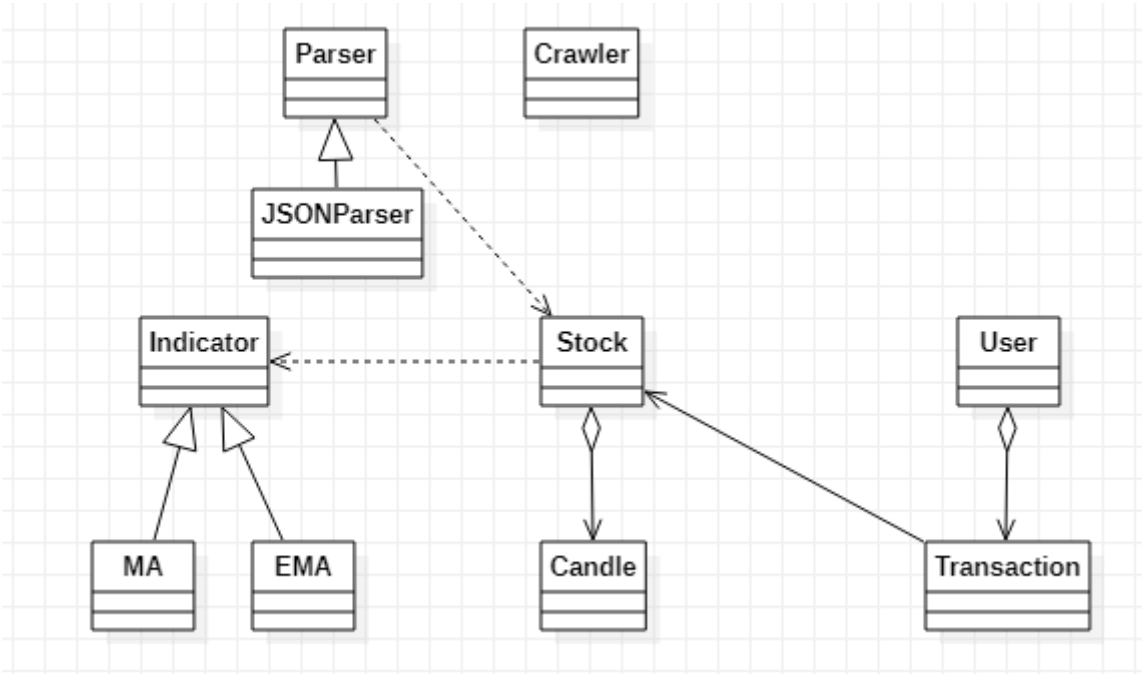
Testiranje rada programa

Prilikom testiranja rada programa moguće je koristiti sajt *tradingview.com/chart*. Cena akcija na ovom sajtu mogu blago odstupati od cena koje se dobijaju sa *finance.yahoo.com*. Pored cena, na ovom sajtu moguće je proveriti tačnost i statističkih indikatora *MA* i *EMA*. Oni se na dijagramu prikazuju u vidu linije, a mogu se uključiti koristeći opciju *Indicators* iz menija. Na slici 4 je plavom linijom prikazan *MA(20)* za akciju AAPL sa dnevnim svećama. Primećuje se da *MA(20)* opisuje trend kretanja cene akcije.

Slika 4 Primer MA(20)

Dijagram klasa

Na osnovu prethodne funkcionalne specifikacije formiran je sledeći dijagram klasa. Dijagram klasa nije detaljan, te ga treba tumačiti kao skicu koja načelno ukazuje na arhitekturu softvera. Studenti mogu da koriste ovaj dijagram kao referencu i, po potrebi, prošire ga da bi ga usaglasili sa eventualnim dopunama specifikacije.



Prilikom implementacije rešenja, obratiti pažnju na objektno orijentisani dizajn i intenzivno koristiti kolekcije i algoritme standardne biblioteke jezika C++ i lambda funkcije gde god je to moguće.

Specifikacija JSON formata

JSON (*JavaScript Object Notation*) je format za razmenu podataka. Lako se parsira i generiše, a čak je i ljudima pogodan za čitanje i pisanje. U pitanju je u potpunosti tekstualni format, nezavisan od bilo kog programskog jezika ili platforme. JSON je izgrađen na osnovu dve strukture:

- Kolekcija parova ključeva i vrednosti - **objekat**, heš tabela, record, struct,...

  Primer objekta je dat u nastavku:

  ```
  {
      "voce": "Jabuka",
      "velicina": "Velika",
      "boja": "Crvena"
  }
  ```

  U vitičastim zagradama se, odvojeni zarezom, navode parovi ključeva i vrednosti. Ključ i vrednost se razdvajaju dvotačkom.

- Uređena lista vrednosti - **niz**, lista, vektor, sekvenca,...

  Primer niza je dat u nastavku:

  ```
  [ "Crvena", "Zelena", "Plava", "Bela"]
  ```

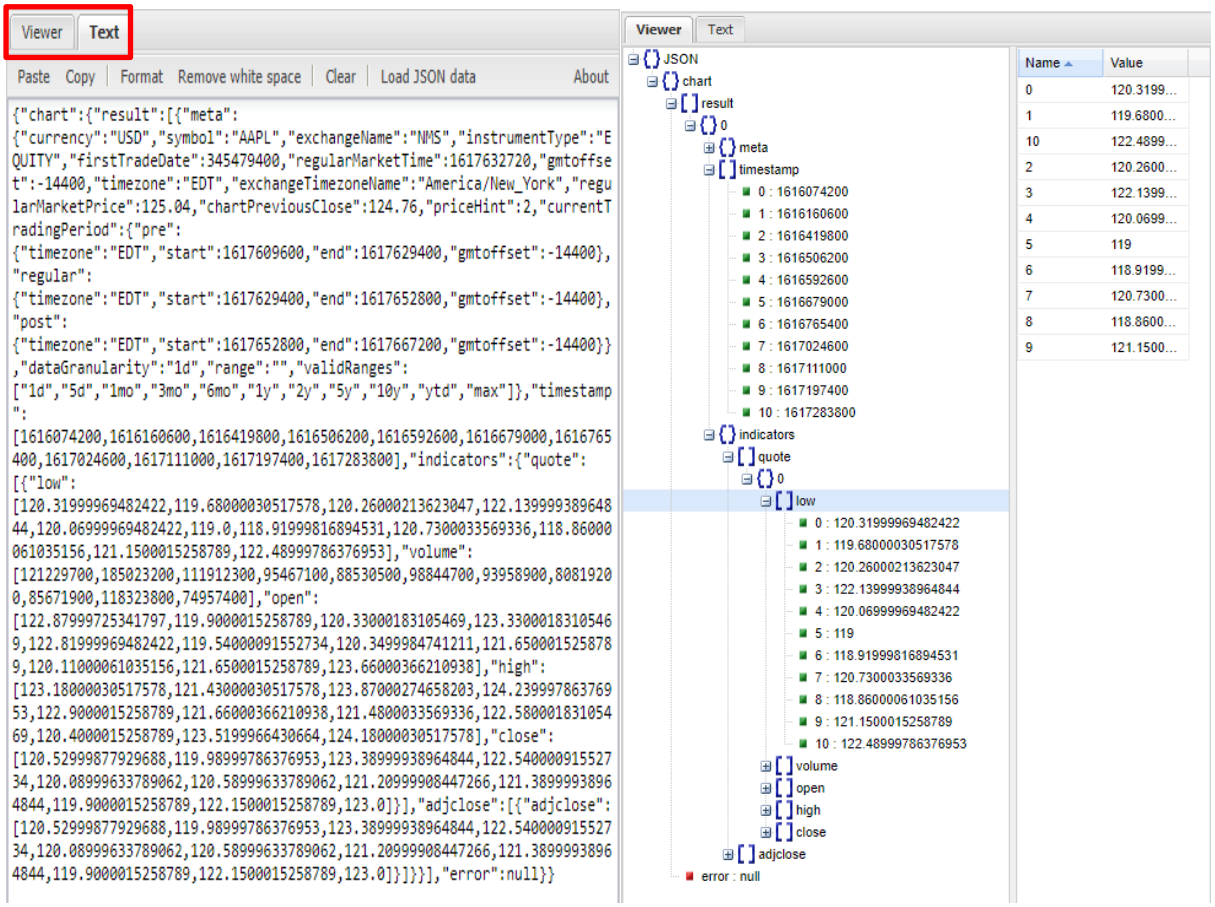
Ove dve strukture se mogu proizvoljno kombinovati čime se dobijaju složene strukture. Primer je dat u nastavku:

```
{ "kviz": {
        "sport": {
            "p1": {
              "pitanje": "Kojim sportom se bavi Novak Djokovic?",
              "odgovori": ["Tenis","Odbojka","Kosarka"],
              "tacan": "Tenis"
            }
        },
        "matematika": {
            "p1": {
                "pitanje": "5 + 7 = ?",
                "odgovori": ["10","11","12","13"],
                "tacan": "12"
            }
        }
    }}
```

Kako biste lakše razumeli JSON format otvorite sledeći link u bilo kom browser-u:

https://query1.finance.yahoo.com/v8/finance/chart/aapl?period1=1616072670&period2=1617531870&interval=1h

Iskopirajte celokupan sadržaj, a zatim posetite sajt http://jsonviewer.stack.hu/. Ovaj sajt nudi lak pregled JSON sadržaja. Nalepite (*Paste*) kopirani sadržaj u tekstualno polje i kliknite na dugme *Viewer* u gornjem levom uglu. Primer upotrebe je prikazan na slici 5.



Slika 5 Prikaz JSON sadržaja

curl biblioteka

curl je open-source biblioteka za prenos podataka. Podržava širok spektar Internet protokola, među kojima i *HTTP* protokol koji je od interesa u ovom projektu. Postupak instalacije biblioteke na *Windows* operativnom sistemu je dat u nastavku:

1. Skinite zip fajl sa sledećeg linka: https://curl.haxx.se/download/curl-7.70.0.zip
2. Raspakujte zip u proizvoljan folder (u daljem tekstu CURL folder)
3. Iz Start menija pokrenite ***Developer Command Prompt for VS 20XX*** (XX u zavisnosti od verzije Visual Studio alata koju koristite)
4. Otkucajte ***cd CURL\winbuild*** (obratite pažnju da se CURL odnosi na folder u koji ste raspakovali zip u koraku 2)
5. Izvršite sledeću komandu (može potrajati): ***nmake /f Makefile.vc mode=static***
6. Otvorite C++ projekat u kome želite da koristite ovu biblioteku u Visaul Studio alatu
7. U ***Solution Explorer***-u kliknite desni klik na naziv projekta, a zatim izaberite opciju ***Properties***
8. U ***VC++ Directories -> Include Directories*** dodajte CURL\builds\libcurl-vc-x86-release-static-ipv6-sspi-winssl\include\
9. U ***VC++ Directories -> Library Directories*** dodajte CURL\builds\libcurl-vc-x86-release-static-ipv6-sspi-winssl\lib\
10. U ***Linker -> Input -> Additional Dependencies*** dodajte ***libcurl_a.lib, Ws2_32.lib, Crypt32.lib, Wldap32.lib*** i ***Normaliz.lib***

Primer upotrebe biblioteke nakon instalacije je dat u nastavku:

```
#include <iostream>
#include <string>
#define CURL_STATICLIB
#include <curl/curl.h>

static size_t WriteCallback(void* contents, size_t size, size_t nmemb, void* userp)
{
    ((std::string*)userp)->append((char*)contents, size * nmemb);
    return size * nmemb;
}

int main(void)
{
    CURL* curl;
    CURLcode res;
    std::string readBuffer;
    curl = curl_easy_init();
    if (curl) {
        curl_easy_setopt(curl, CURLOPT_URL,
"https://query1.finance.yahoo.com/v8/finance/chart/tsla?period1=1617272670&period2=161
7531870&interval=1d");
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteCallback);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, &readBuffer);
        res = curl_easy_perform(curl);
        curl_easy_cleanup(curl);
    }

    std::cout << readBuffer;
    return 0;
}
```