Sveučilište u Rijeci – Odjel za informatiku

Objektno programiranje

Projektna dokumentacija za

OIRI Notepad

Antonio Janach

Rijeka, veljača 2022.

Sadržaj

1. Uvod

U ovoj projektnoj dokumentaciji bit će prikazan proces izrade desktop aplikacije koristeći wxWidgets. To je programski alat za pisanje desktop ili mobilnih aplikacija s grafičkim sučeljima (GUI), omogućuje stvaranje aplikacija na različitim platformama. Platforme koje wxWidgets pokriva su Microsoft Windows, Mac OS, Linux. To omogućuje programerima da ne moraju istu aplikaciju prilagođavati posebno za svaki od operacijskih sustava i pisati zasebno kôd, već wxWidgets omogućava izradu aplikacije za navedene operacijske sustave koristeći isti kôd s minimalnim promjenama ili bez njih. Također, bitno je napomenuti da se wxWidgets licenca temelji na LGPL-u. To znači da je s ovom licencom moguće razviti komercijalni proizvod bez plaćanja.

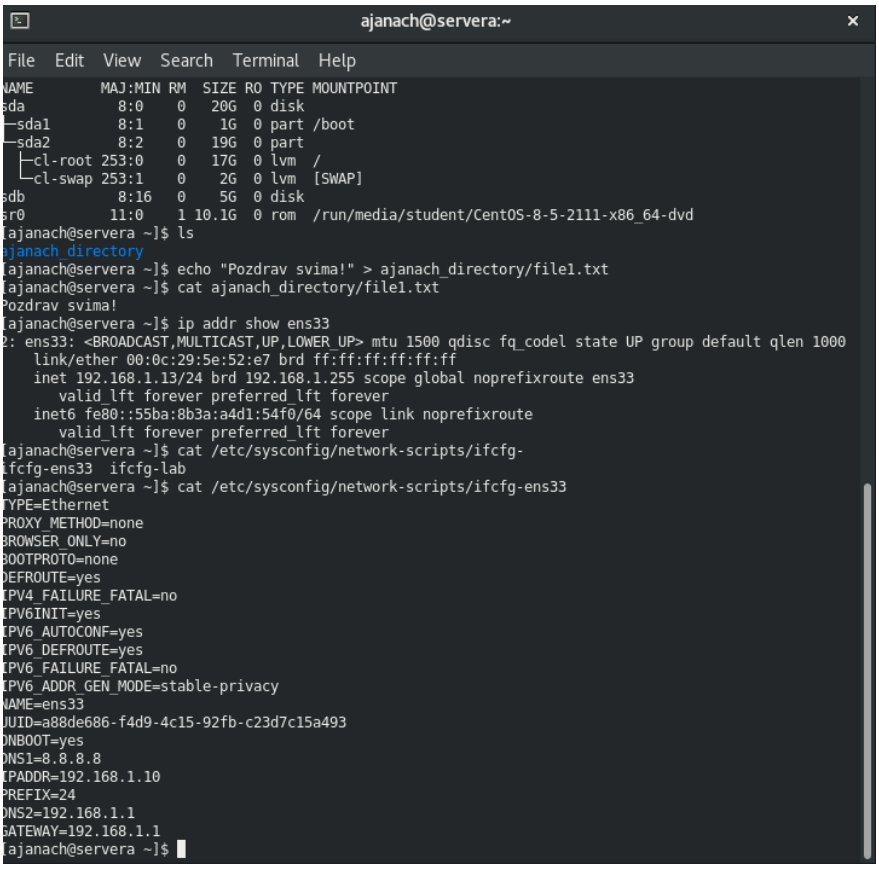
Za početak bit će objašnjeni ključni pojmovi koji se već spominju u samoj temi ovog rada. Što je to sučelje, što je grafičko sučelje, što je C++ i kako se koriste razni razvojni okviri. Također, bit će objašnjeno kako je primijenjena razvojna okolina, koji su korišteni alati te će detaljno biti opisano programsko rješenje razvoja navedene aplikacije, i bit će opisani glavni dijelovi programskog kôda. Kroz klasni dijagram vizualno će biti prikazano koje klase wxWidgets programskog alata se koriste. Zatim će biti dane upute za build programa zajedno sa osnovnim uputama korištenja same aplikacije.

Zahtjevi izrade aplikacije za uređivanje teksta temelje se na osobnom odabiru, kao referenca korištene su funkcionalnosti srodnog tekstualnog uređivača koji dolazi s instaliranim Microsoft Windows operacijskim sustavom. Uređivač teksta koji se nalazi na Windows operacijskom sustavu zove se Notepad, vrlo stara aplikacija koja je desetljećima prisutna Windows korisnicima.

Glavna svrha i razlog odabira izrade navedene aplikacije jest da jasno prikazuje koncept objektno orijentirane paradigme u C++ programskom jeziku. Također, izrada ovog projekta čini idealnim za učenje i korištenje wxWidgets programskog alata s grafičkim sučeljem.

2. Vrste korisničkih sučelja kod aplikacija

U ovom području informatike definicija sučelja glasi da je sučelje veza između dvije ili više odvojenih komponenata u računalnom sustavu preko koje oni razmjenjuju informacije. Sučelje tako može povezivati softver, hardver, periferijske uređaje, ljude. Tako kaže Hookway (Hookway, 2014). U ovome radu naglasak je na korisničko sučelje (eng. *User interface*), odnosno mjesto gdje se obavlja razmjena informacija između računala i čovjeka. Opet, postoje različiti tipovi korisničkih sučelja i već se desetljećima razvijaju. Kako je rečeno, naglasak je na razvoju najmodernijeg tipa, GUI (eng. *graphical user interface*), odnosno grafičko korisničko sučelje.



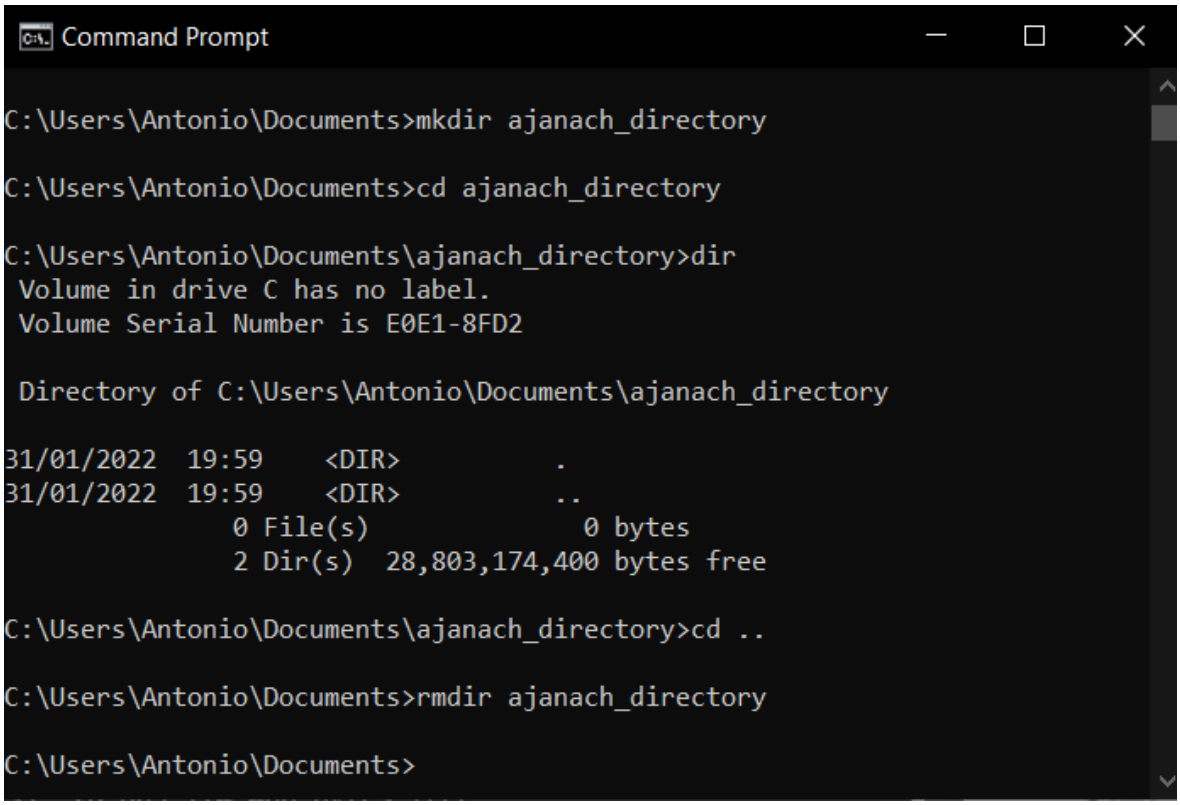
Slika 1: primjer sučelja naredbenog retka na Linux operacijskom sustavu

2.1. Command-line interface (CLIs)

Sučelje naredbenog retka je sučelje koje održava interakciju s računalnim programom tako da korisnik govori programu što da radi u obliku linije teksta, odnosno naredbe. Program koji odrađuje interakciju naziva se interpreter naredbenog retka, ili Shell.

Ovakvo sučelje je bilo primarni tip interakcije s većinom računalnih sustava u kasnijim 1960-ima godina, a nastavili su se koristiti do danas.

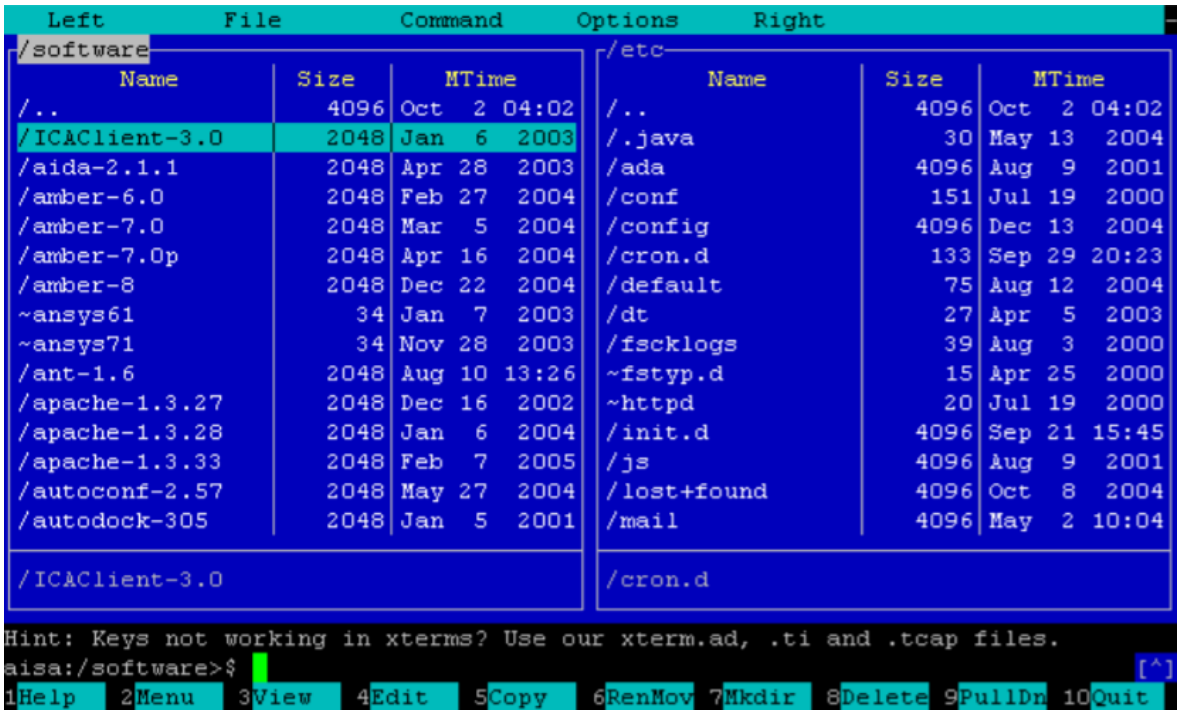
Sučelje naredbenog retka koristi se kada postoji veliki broj naredbi ili upita sa zasebnim opcijama. Tada je lakše i brže reći programu što da radi, nego da se implementiraju sve funkcije u grafičko sučelje. Najčešće je to primjer s operacijskim sustavima i njihovim naredbenim recima. Mogu se koristiti i u sustavima koji nemaju dovoljno resursa da pruže grafičko sučelje. (What is CLI, 2022.)



Slika 2: primjer rada s direktorijima u Windows naredbenom retku

2.2. Text based user interface ili SAA

1985 godine, na početku razvoja Microsoft Windows-a i ostalih grafičkih korisničkih sučelja, IBM je kreirao Systems Application Architecture (SAA) standard koji je uključivao Common User Access (CUA). CUA je ono što danas koristimo u Windows OS-u, također je tada većina Windows Console Applications koristila taj standard. Taj standard je definirao da sustav izbornika treba biti na vrhu prozora, prikaz statusa na dnu aplikacije i da tipke prečice ostanu na istoj funkcionalnosti. (Richard, 2022.)



Slika 3: primjer tekstualnog sučelja (Powell, 1997.)

2.3. Graphical user interface

Grafičko korisničko sučelje je oblik korisničkog sučelja koje omogućava korisnicima interakciju s elektroničkim uređajima pomoću ikona i vizualnih indikatora.

1963. godine razvijen je Sketchpad kojim se smatra prvi program sa „grafikom“. U 1970-ima proširuju se ideja o korisničkom sučelju u Xerox-u te počinju koristiti GUI kao glavno sučelje iz kojih nastaju većina današnjih grafičkih korisničkih sučelja. (Shotts, 2019)

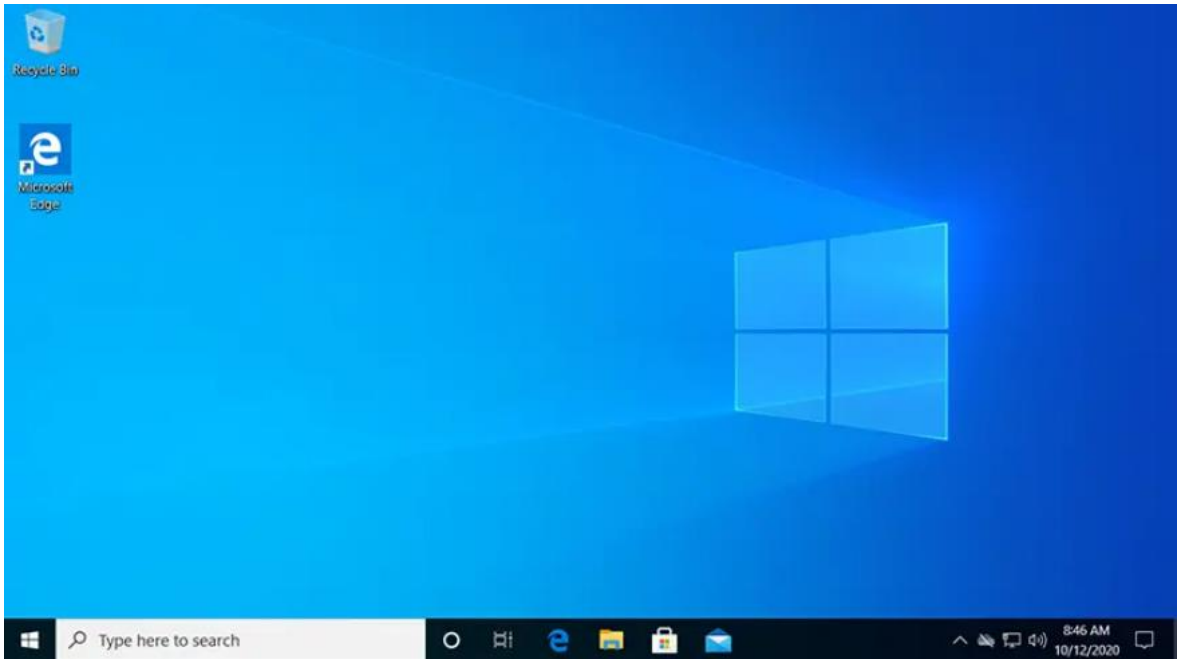
Takav sustav sastojao se od prozora, izbornika, radio gumbova i check box-ova. Alternativni akronim WIMP označava *windows, icons, menus, pointing devices*. Godinama se razvijalo ovakvo sučelje dok sredinom 1980-ih Apple nije popularizirao takva sučelja.

Slika 4: 1983 izdana je Apple Lisa (GUI) (Apple, 2008.)

Još uvijek kritičari nisu podržavali koncept grafičkih sučelja navodeći da se javljaju problemi s hardware-om i kompatibilnim softverom. Godinu dana kasnije, 1984, Apple reklamom nagovara javnost da razmišlja o takvom sučelju kao svoje, osobno računalo.

Tako nastaje Windows 95 i ostvaruje veliki uspjeh, te ubrzo postaje najpopularniji operacijski sustav za osobna računala.

Slika 5: Windows 95 (GUI) (Microsoft, Wikipedija, 2021.)



Slika 6: Windows 10 (GUI) (Microsoft, 2021.)

3. Razvojni okvir

U računalnom programiranju, razvojni okvir je apstrakcija u kojem se softver koji pruža općenitu funkcionalnost mijenja korisničkim kodom i tako se svrha softvera sužava, odnosno postoje specifičnija. To je univerzalno okruženje koje pruža određenu funkcionalnost kao dio veće platforme za razvoj programa, aplikacija i rješenja. Takvi okviri mogu sužavati potporne programe, biblioteke, skupove alata, API (eng. *Application programmable interface*) koji spajaju komponente u razvoj projekata ili sustava.

Postoje ključne razlike koje razvojne okvire čine razvojnim okvirima, za razliku od običnih biblioteka. U okviru, tijek rada kontrolira sami razvojni okvir, a ne korisnik. Korisnik može proširiti razvojni okvir prepisivanjem (eng. *override*) ili dodavanjem kôda određene funkcionalnosti. Kôd razvojnog okvira se općenito ne mijenja. (W3Schools, 2022)

4. Priprema okoline i korišteni alati

U ovom poglavlju bit će prikazano kako je pripremljena okolina za rad, koji programski jezik se koristi, detaljnije od wxWidgets programskom alatu i koji tekstualni editor je korišten.

4.1. C++

C++ je opće namjenski, objektno orijentirati programski jezik koji je stvorio Bjarne Stroustrup kao ekstenziju na C jezik. C++ obuhvaća i visoke i niske značajke programskih jezika, te se zbog toga smatra jezikom srednje razine. U početku zvan „C s klasama“ zbog sadržavanja svih svojstva C jezika, te nadodavanja koncepata klasa.

Jedno od glavnih svojstva C++ jezika je zbirka unaprijed definiranih klasa, koji su tipovi podataka koje je moguće instancirati više puta i omogućuje deklaraciju korisničko definiranih klasa. Same klase se dalje mogu prilagođavati funkciji svojih članova radi implementacije određenih funkcionalnosti. C++ omogućava preopterećenje različitih operatora kao što su operatori za aritmetiku, usporedbu, logički operatori ili operatori za manipuliranje bitovima. Uvode se bitni koncepti polimorfizma i nasljeđivanja, kao i virtualne i prijateljske funkcije, predlošci i *namespace*-ovi. (Tutorialspoint, 2022)

Nasljeđivanje je jedan od najvažnijih koncepata objektno orijentiranog programiranja. Nasljeđivanje nam omogućava definiranje klase u smislu druge klase, što olakšava stvaranje i održavanje aplikacije. Također pruža priliku za ponovnu upotrebu funkcionalnosti koda i brzu implementaciju.

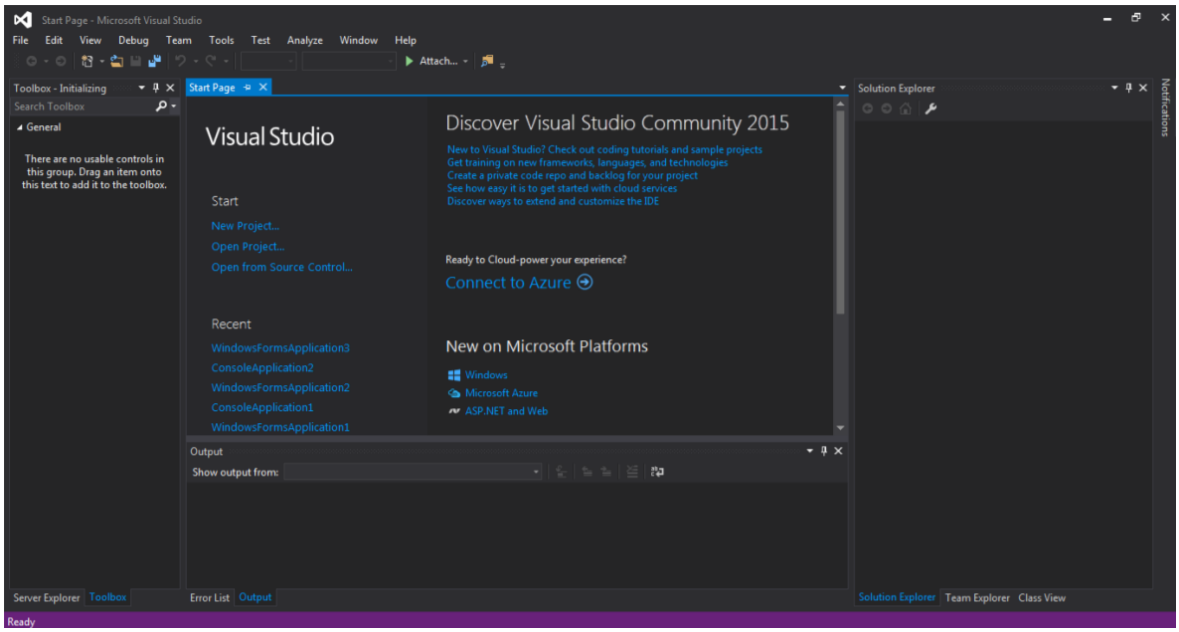
Kada se stvara klasa, umjesto pisanja potpuno novih članova podataka i funkcija članova, programer može odrediti da nova klasa treba nasljeđivati članove postojeće klase. Postojeća se klasa naziva osnovna klasa, a nova klasa, koja je izvedena, naziva se izvedenom klasom.

4.2. Visual studio

Može se reći kako Microsoft nudi mnoge alate i tehnologije koji služe korisnicima u izradama aplikacija, ali odabirom Microsoft tehnologija na samom početku korisnici baš i nemaju potrebno znanje za izradu funkcionirajući aplikacije. U posljednje vrijeme Microsoft

se puno više trudi izdavati vodiče za korištenje njihovih proizvoda i razvijati svoje alate u skladu s potrebama i sugestijama mnogih programera. Microsoft redovito izdaje vodiče u obliku videa koji detaljno opisuju mogućnost, izmjenu sadržaja te nove mogućnosti u novijim inačicama alata koje izdaju.

Microsoft je razvio Visual Studio kao proizvod idealan za programere kako bi im omogućio izrađivanje mnogih vrsta aplikacija. Prema (Microsoft, Wikipedia, 2022), Microsoft Visual Studio koristi se za izradu *desktop* aplikacija, web aplikacija ili mobilnih aplikacija. To je ujedno i jedan od najraširenijih alata za izradu aplikacija na Windows operacijskim sustavima. Prednost Visual Studia je to što je dosta fleksibilan i omogućuje korisnicima jednostavan razvoj mobilnih aplikacija, ali on zahtjeva puno više vremena za savladavanje svih njegovih mogućnosti. Postoje tri glavne inačice Visual Studio integriranog razvojnog okruženja: Visual Studio, Visual Studio Code i Visual Studio Online. Također postoji više inačica programa: Visual Studio Community (besplatna inačica dostupna na korištenje za akademske svrhe ili individualni razvoj), Visual Studio Professional (plaćena inačica s raznim korisnim alatima za razvoj namijenjena manjim timovima ili pojedinačnim developerima) te Visual Studio Enterprise (plaćena inačica za poduzeće i timove bilo koje veličine s naprednim alatima za razvoj i najkompleksnijih rješenja). Cilj svake inačice Visual Studia je pružiti bogato razvojno okruženje svim programerima na globalnoj razini na bilo kojoj platformi. Visual Studio nudi bogat izbor razvojnih jezika. Trenutno programeri mogu razvijati aplikacije u jezicima Visual Basic, C#, PHP, Objektni C, JavaScript i Visual C++.



Slika 7: Korisničko sučelje Microsoft Visual Studia

4.3. WX Widgets

wxWidgets korišten je za izradu sučelja aplikacije OIRI Notepad koja će biti opisana kasnije. To je programski alat za izradu aplikacija na sustave kao što su Microsoft Windows, Mac OS X i Linux/Unix koristeći programski jezik C++, no moguće je preko omotača koristiti i druge jezike, na primjer Erlang, Perl, Ruby, Python ili Javu. wxWidgets je otvorenog kôda, posjeduje opširnu dokumentaciju s primjerima te pruža dobru podršku za otkrivanje pogrešaka.

Glavna klasa svake aplikacije koja koristi wxWidgets mora naslijediti klasu `wxApp` i nadjačati njenu `OnInit()` metodu pomoću koje se aplikacija inicijalizira. U toj klasi postavljaju se svojstva aplikacije i implementira se sustav za upravljanje događajima.

Nasljeđivanjem klase `wxFrame` stvara se vlastiti prozor koji može sadržavati izbornik, alatnu traku i statusnu traku. Unutar te klase definira se tablica događaja na koje aplikacija reagira (DECLARE_EVENT_TABLE()).

Manji prozori se izrađuju nasljeđivanjem klase `wxDialog`.

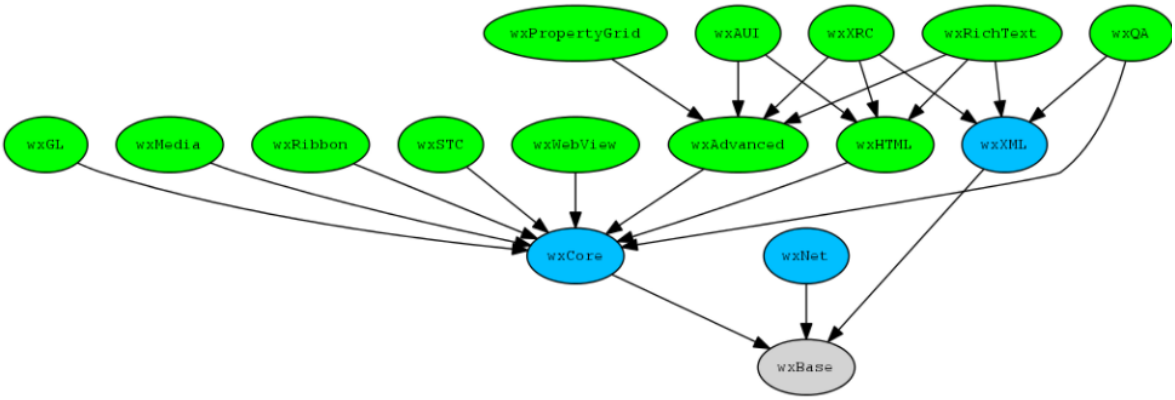
Za izradu padajućeg izbornika služi klasa `wxMenu` dok za složenije izbornike sastavljene od više objekata tipa `wxMenu` koristi se klasa `wxMenuBar`. Alatna traka kreira se pomoću klase wxToolbar, a gumbe pomoću `wxButton`.

wxWidgets se sastoji od još mnoštva klasa koje se koriste u razne svrhe, u ovom kratkom pregledu navedene su samo najosnovnije.

4.3.1. wxWidgets biblioteke

Počevši od verzije 2.5.0 wxWidgeti se mogu izgraditi kao jedna velika biblioteka (ovo se zove *monolithic build*) ili kao nekoliko manjih biblioteka (odnosno *multilib build*). *Multilib build* je zadana postavka wxWidgets-a.

Biblioteka wxWidgets-a podijeljena je na biblioteke koje su ukratko opisane u nastavku.



Slika 8: dijagram biblioteka i prikaz ovisnosti između njih [13 dodati literaturu]

Potrebno je imati na umu da strelice označavaju odnos „ovisi od“ i da sve plave biblioteke ovise o wxBase biblioteci (nisu dio GUI biblioteke), a sve zelene biblioteke ovise o wxCore biblioteci (one su dio GUI biblioteke).

wxBase

Svaka aplikacija wxWidgets mora se povezati s ovom bibliotekom. Sadrži obvezne klase o kojima wxWidgets kôd ovisi (npr. wxString) i klase prenosivosti koje izdvajaju razlike između platformi. wxBase se može koristiti za razvoj aplikacija u konzolnom načinu rada, ne zahtijeva nikakve GUI biblioteke.

wxCore

Biblioteka sadrži osnovne GUI klase. Sve wxWidgets GUI aplikacije moraju se povezati s ovom bibliotekom, samo aplikacije u konzolnom načinu rada ne.

Zahtijeva: wxBase

wxGL

Ova biblioteka sadrži klasu wxGLCanvas za integraciju OpenGL biblioteka s wxWidgetima. Za razliku od svih ostalih, ova knjižnica nije dio monolitne biblioteke, uvijek je izgrađena kao zasebna biblioteka.

Zahtijeva: wxBase, wxCore

wxHTML

Jednostavni HTML renderi sadržani su u ovoj biblioteci, kao wxHtmlHelpController, wxBestHelpController i wxHtmlListBox.

Zahtijeva: wxCore, wxBase

wxMedia:

Razne klase vezane uz multimediju. Trenutno ova biblioteka sadrži samo wxMediaCtrl, ali u budućnosti će biti dodano više klasa.

Zahtijeva: wxCore, wxBase

wxNet:

Klase za pristup mreži. wxSocket klase, wxSocketOutputStrem i wxSocketInputStream, IPC klase zasnovane na socket-ima, wxURL, wxInternetFSHandler.

Zahtijeva: wxBase

wxQA:

Ova biblioteka sadrži dodatne klase za osiguranje kvalitete. Trenutno sadrži wxDebugReport i povezane klase.

Zahtijeva: wxXML, wxCore, wxBase

wxRibbon:

Sadrži biblioteku komponenti korisničkog sučelja vrpce.

Zahtijeva: wxCore, wxBase

wxRichText;

Sadrži generičku funkciju kontrole obogaćenog teksta.

Zahtijeva: wxAdvanced, wxHTML, wxXML, wxCore, wxBase

wxSTC:

STC (Styled Text Control), komponenta za uređivanje izbornog koda.

Zahtijeva: wxCore, wxBase

wxWebView:

Zahtijeva: wxCore, wxBase

wxXML:

Sadrži jednostavne klase za raščlanjivanje XML dokumenata.

Zahtijeva: wxBase

wxXRC:

Ova knjižnica sadrži klasu wxXmlResource koja omogućuje pristup XML datoteka resursa u XRC formatu

Zahtijeva: wxAdvanced, wxHTML, wxXML, wxCore, wxBase

wxAdvanced:

Napredne ili rijetko korištene GUI klase.

Zahtijeva: wxCore i wxBase

wxAui:

Sadrži biblioteku za priključivanje naprednog korisničkog sučelja.

Zahtijeva: wxAdvanced, wxHTML, wxXML, wxCore, wxBase

5. Programsko rješenje OIRI Notepad aplikacije

U ovome poglavlju bit će detaljno objašnjen opis strukture programa, opis glavnih dijelova aplikacije, odnosno klasa, bit će spomenute biblioteke i zaglavlja koja se koriste i čemu služe. Također, za OIRI Notepad aplikaciju bit će prikazan dijagram klasa.

5.1. Korištene biblioteke za razvoj OIRI Notepad aplikacije

Biblioteke koje su korištene za razvoj OIRI Notepad aplikacije su:

- wxBase
- wxCore
- wxAdvanced
- string - standardna biblioteka C++ programskog jezika

Detaljni opis wxWidgets korištenih biblioteka može se pronaći u poglavlju 4.3.1

5.2. Opis glavnih dijelova aplikacije

Aplikacija OIRI Notepad sastoji se od dvije klase imena Notepad i MainApp. Detaljne funkcije klasa bit će opisane daljnje u ovom dijelu poglavlja. Aplikacija mora imati uključene biblioteke kako bi određene funkcionalnosti radile. Također, aplikacija se sastoji od makronaredba koje su vezane za tablicu događaja i implementacije same aplikacije, odnosno "main" funkcije.

5.2.1. Uključivanje datoteka zaglavlja wxWidgetsa

Prvo, je naravno potrebno uključiti datoteke zaglavlja wxWidgetsa. Za svaki napisani program korištenjem wxWidgets programskog potrebno je uključiti globalno zaglavlje imena `wx/wx.h`. To zaglavlje uključuje većinu uobičajeno potrebnih zaglavlja (iako ne sva jer jednostavno ima previše wxWidgets zaglavlja koja je moguće uključiti). Ostala uključena zaglavlja u programskom kôdu koriste se kako bi napredne funkcionalnosti bile dostupne za korištenje u samoj aplikaciji.

Zaglavlja koja su korištena tijekom razvoja OIRI Noteped tekstualnog editora su:

wx/richmsgdlg.h >

Ovo zaglavlje, dodaje mogućnost korištenja potvrdnog okvira. Korisno je za implementaciju dijaloških okvira vrste „Ne pitaj me više“ i dodatnog teksta objašnjenja koji je u početku sažet i nije prikazan korisniku, ali se može proširiti kako bi se prikazalo više informacija.

wx/printdlg.h >

Ovo zaglavlje predstavlja uobičajene dijaloške okvire za postavljanje ispisa sadržaja na papir koristeći pisač.

wx/string.h >

Zaglavlje korišteno za prosljeđivanje ili primanje teksta korištenjem wxWidgetsa. Odnosno, zaglavlje služi za manipulaciju teksta.

wx/fdrepdlg.h >

zaglavlje korišteno za dijalog koji se koristi da omogući korisniku da traži neki tekst i eventualno ga zamijeni nečim drugim.

wx/numdlg.h >

Zaglavlje koje se koristi za prikaz dijaloškog okvira u kojem se od korisnika traži numerički unos.

wx/fontdlg.h >

Zaklavlje koje se koristi za prikaz dijaloškog okvira za odabir fonta.

wx/aboutdlg.h >

Zaglavlje koje se koristi za prikaz informacija o programu, kao što su naziv, verzija, autorska prava i tako dalje, kao i popis programera, pisca dokumentacije…

wx/msgdlg.h >

Zaglavlje koje se koristi za prikaz dijaloškog okvira koji prikazuje poruku, s opcijama odabira na OK, YES, NO i CANCLE gumbe.

String >

Standardna biblioteka C++ programskog jezika koja predstavlja niz znakova. Korištena za pretvaranje numeričkog tipa podataka u znakovni tip podataka (long u string).

Prikaz svih uključenih zaglavlja koja su korištena u programskom kôdu:

```
//uključena zaglavlja:
#include <wx/wx.h>
#include <wx/richmsgdlg.h>
#include <wx/printdlg.h> //wxCore
#include <wx/string.h> //wxCOre
#include <wx/fdrepdlg.h> //wxCore
#include <wx/numdlg.h> //wxCore
#include <wx/fontdlg.h> //wxCore
#include <wx/aboutdlg.h> //wxAdvanced
#include <wx/msgdlg.h> //wxCore
#include <string> //string
```

Kôd 1: prikaz uključenih zaglavlja

Ovime završava poglavlje koje opisuje koja su zaglavlja uključena, a koja su temelj za daljnje korištenje funkcija same aplikacije.

5.2.2. Klasa Notepad

Klasa imena Notepad stvara glavni prozor tako da se klasa izvede iz wxFrame-a i da joj se u konstruktoru daju izbornik i statusna traka. Također, svaka klasa koja želi odgovor na bilo koji događaj kao što su klikovi mišem ili poruke s izbornika ili gumba mora deklarirati tablicu događaja koristeći makronaredbu, ova makronaredba uobičajeno se deklarira na kraj klase.

Primjer sintakse definiranja obrade događaja u klasi izgleda ovako:

```
wxDECLATE_EVENT_TABLE()
```

Kôd 2: prikaz definiranja obrade događaja u klasi

Način reagiranja na takve događaje mora se obraditi korištenjem *handlerima*. U ovom slučaju ova klasa programskog kôda reagira na 28 događaja.

Kako bi se moglo reagirati na događaj, događaju je potrebno dati jedinstveni identifikator koji se može definirati kao *const* varijabla, također može biti dio *enum* (nabrajanje)

elemenata. Za svaku metodu koja se nalazi u navedenoj klasi potrebno je definirati identifikator.

U navedenoj klasi samo je zadani konstruktor javan, a taj konstruktor omogućuje kreiranje same okvira, trake izbornika, izbornika za svaku karticu posebno, statusne trake, zatim postavljanje ikone programa i postavljanje zadanog naslova okvira. Zadani konstruktor zapravo izvodi zadanu operaciju za konstruktor wxFrame.

U privatnom dijelu klase kreirani su objekti i jedna varijabla tipa bool. Također, u privatnom dijelu klase nalaze se metode, *enum* elementi i deklaracija tablice događaja. U daljnjem dijelu ovog poglavlja nalazi se popis metoda klase i što svaka metoda radi, kôd svake metode može se pogledati u poglavlju „Prilog: programski kod“ gdje se nalazi cijeli kôd programa. Kôd za svaku metodu ovdje nije prikazan jer je sadržajno pre velik. I za kraj bit će prikazani enum elementi.

Popis metoda klase Notepad:

1. OnFileNew
    - Brisanje tekstualnog sadržaja koji se nalazi u okviru za unos teksta. Postavljanje naslova okvira na „Untilted“ i poništava se putanja datoteke ne prazno.
2. OnFileNewWindow
    - Otvara se novi okvir tekstualnog editora, odnosno poziva se zadani konstruktor.
3. OnFileOpen
    - Otvara se novi dijaloški okvir za otvaranje datoteka. Odabirom na datoteku koja ima ekstenziju .txt, .cpp ili odabirom na bilo koji tip datoteke i pritiskom na gumb „Open“ otvara se datoteka i prikazuje se u okviru za unos teksta. Zatim se mijenja ime naslova okvira prema imenu datoteke koja je otvorena, postavlja se putanja datoteke u varijablu na za to predviđeno mjesto.
4. OnFileSave
    - Metoda se sastoji od dva if uvjeta, tako da, ako je korisnik aplikacije prethodno pohranio svoj dokument ili ga je otvorio to znači da je aplikacija pohranila putanju datoteke i aplikacija može samo pohraniti modifikaciju iste datoteke na istu putanju. Ali, ako korisnik nije prethodno spremao datoteku ili ju otvorio tada aplikacija u sebi nema spremljenu putanju datoteke, zbog

toga se otvara dijaloški okvir za spremanje datoteke gdje korisnik navede putanju spremanja, pritisne gumb „Save“ i datoteka se sprema. Metoda pohranjuje putanju datoteke u varijablu i postavlja naslov okvira u ime koje je dodijeljeno datoteci.

5. OnFileSave
    - Otvara se novi dijaloški okvir za spremanje datoteka. Korisnik pomoću dijaloškog okvira za spremanje datoteke odabire mjesto spremanja, upisuje ime datoteke i unutar dijaloškog okvira korisnik odabire tip tekstualne datoteke koji želi spremiti. Tip datoteka koje korisnik može spremiti su ekstenzija .txt, .cpp i *.* (bilo koji tip datoteke). Kad je korisnik odabrao navedeno i pritisnuo na gumb „Save“, sprema se sadržaj u datoteku, mijenja se naslov okvira u ime koje je korisnik dao datoteci i sprema se putanja spremljene datoteke u varijablu na za to predviđeno mjesto.
6. OnFilePageSetup
    - Otvara se novi dijaloški okvir za postavljanje stranice.
7. OnFilePrint
    - Otvara se dijaloški okvir za ispis sadržaja na papir.
8. OnFileExit i OnXButton
    - Jedna i druga metoda rade isto samo jedna odgovara na događaj kada je pritisnut „Exit“ gumb iz izbornika, a druga metoda odgovara kada je pritisnut „X“ gumb okvira prozora. Metode prvo provjeravaju da li je bilo kakav sadržaj upisan na mjesto koje je predviđeno za upisivanje teksta. Ako sadržaj nije upisan od strane korisnika program se gasi bez da informira korisnika o spremanju sadržaja. Ako je korisnik upisao sadržaj, otvara se info okvir gdje aplikacija korisnika informira o spremanju sadržaja kojeg je unio u aplikaciju. Korisnik iz info okvira može odabrati tri opcije, odnosno pritisnuti na tri gumba. Pritiskom na „Yes“ gumb korisnik može spremiti sadržaj u datoteku i nakon toga se aplikacija gasi. Pritiskom na „No“ gumb korisniku aplikacija ne sprema sadržaj u datoteku, već se aplikacija gasi. Pritiskom na „Cancel“ gumb korisnika aplikacija vraća nazad u početni okvir kako bi nastavio uređivati svoj sadržaj, isto vrijedi i, ako korisnik pritisne na „X“ gumb.
9. OnEditUndo

- Metoda poništava dodano, a ponovnim pritiskom vraća poništeno. Ova metoda sprječava od eventualnih grešaka ili nenamjernog brisanja sadržaja.

10. OnEditCut
    - Metoda izrezuje označeni tekst i taj izrezani tekst se sprema u među spremnik. Korištenjem metode OnEditPaste taj tekst se iz među spremnika lijepi na mjesto gdje je korisnik prethodno postavio pokazivač za pisanje teksta.
11. OnEditCopy
    - Metoda kopira označeni tekst i taj označeni tekst se sprema u među spremnik. Korištenjem metode OnEditPase taj tekst se iz među spremnika lijepi na mjesto gdje je korisnik prethodno postavio pokazivač za pisanje teksta.
12. OnEditPaste
    - Metoda iz među spremnika lijepi tekst na mjesto gdje je korisnik prethodno postavio pokazivač za pisanje teksta.
13. OnEditDelete
    - Metoda briše označeni tekst.
14. OnEditFind
    - Otvara se novi dijaloški okvir za traženje riječi.
15. OnEditReplace
    - Otvara se novi dijaloški okvir za zamjenu riječi.
16. OnEditGoto
    - Otvara se novi dijaloški okvir gdje korisnik može unjet prirodne brojeve i nulu, aplikacija ne dozvoljava unos svih cijelih brojeva, odnosno negativne brojeve. Kada korisnik unese broj i pritisne gumb „OK“ tada aplikacija korisnika vodi na upisanu liniju teksta.
17. OnEditSelectAll
    - Metoda označava cijeli sadržaj koji je korisnik unesao u aplikaciju.
18. OnEditTimeDate
    - Metoda ispisuje trenutni datum, vrijeme i godinu na mjesto gdje je korisnik postavio pokazivač za unos teksta.
19. OnFormatAutoLine

- Metoda postavlja tekst u jednu liniju i stvara se horizontalni klizač.

20. OnFormatWordWrap
    - Metoda prelama riječi tako da stanu u okvir aplikacije.
21. OnFormatFont
    - Otvara se novi dijaloški okvir za postavljanje fonta.
22. OnViewStatBar
    - Ako je statusna traka uključena tada metoda sakriva statusnu traku, ako je statusna traka isključena tada aplikacija uključuje statusnu traku.
23. OnHelpAbout
    - Otvara se info okvir koji sadrži informacije o imenu programa, verziji programa, opisu programa, *copyright-u* i prikazu imena i prezimena o autoru aplikacije.
24. OnUpdateUIUndo, OnUpdateUICut, OnUpdateUICopy i OnUpdateUIPaste
    - Metode koje se pozivaju u aplikaciji za ažuriranje navedenih elemenata korisničkog sučelja. Bez događaja ažuriranja korisničkog sučelja, aplikacija se mora potruditi provjeriti/poništiti, onemogućiti/omogućiti, prikazati/sakriti i postaviti tekst. Na ovaj način se gleda stanje aplikacije.

Za svaku navedenu metodu implementirano je da se na statusnoj traci na 1.5 ili 2 sekunde ispisuje status koji odgovara navedenoj metodi, kada se događaj koji poziva metodu izvrši ponovno se pojavljuje zadani status.

Prikaz enum elemenata po identifikacijskim brojevima:

```
enum MenuControls {
    idFileNew = 1000,
    idFileNewWindow = 1001,
    idFileSave = 1002,
    idFileSaveAs = 1003,
    idFilePageSetup = 1004,
    idFilePrint = 1005,
    idEditUndo = 1006,
    idEditCut = 1007,
    idEditCopy = 1008,
    idEditPaste = 1009,
    idEditDelete = 1010,
```

```
        idEditFind = 1011,
        idEditReplace = 1012,
        idEditGoto = 1013,
        idEditSelectAll = 1014,
        idEditTimeDate = 1015,
        idFormatAutoLine = 1016,
        idFormatWordWrap = 1017,
        idFormatFont = 1018,
        idViewStatBar = 1019,
        idHelpAbout = 1020,
        idFileOpen = 1021, idFileExit
    };
```

Kôd 3: prikaz enum elemenata

Enum elementi se nalaze u privatnom dijelu navedene klase zajedno s deklaracijom tablice događaja.

5.2.3. Klasa MainApp

U praksi svaka aplikacija koja je pisana koristeći wxWidgets treba definirati novu klasu izvedenu iz wxApp klase. Nadjačavanjem wxApp onInit() virtualne metode program se može inicijalizirati, odnosno stvoriti glavni prozor aplikacije.

Tijekom razvoja OIRI Notepad aplikacije kreirana je klasa imena MainApp koja je izvedena iz wxApp klase.

```
class MainApp : public wxApp {
public:
    virtual bool OnInit() {
        Notepad* main = new Notepad();
        main->Show(true);
        return true;
    }
};
```

Kôd 4: prikaz MainApp klase

5.2.4. Tablica događaja

Kao i kod svih ostalih GUI okvira, kontrola tijeka u wxWidgets aplikacijama temelji se na događajima. Program obično izvodi većinu svojih radnji kao odgovor na događaje koje generira korisnik.

Ti događaji mogu pokrenuti izravnom upotrebom ulaznih uređaja (kao što su tipkovnica, miš, joystick).

Postoje dva glavna rukovanja događajima u wxWidgetima. Jedan od njih je korištenje makronaredbe tablice događaja. Drugi način rukovanja događaja je poziv *event handler-a*. U razvoju navedene aplikacije korišteno je rukovanje događajima korištenjem makronaredbe tablice događaja.

Primjer sintakse makronaredbe tablice događaja izgleda definira se na sljedeći način:

```
//Početak tablice događaja
wxBEGIN_EVENT_TABLE (MyFrame, wxFrame)
    EVT_MENU( wxID_EXIT , MyFrame::OnExit)
    EVT_MENU(DO_TEST, MyFrame::DoTest)
    EVT_SIZE(MyFrame::OnSize)
    EVT_BUTTON(BUTTON1, MyFrame::OnButton1)
wxEND_EVENT_TABLE () //Kraj tablice događaja
```

Kôd 5: izgled sintakse makronaredbe tablice događaja

Bitno je primijetiti da je potrebno spomenuti metodu koja se želi koristiti za rukovanje događajima u definiciji tablice događaja. Zatim, nužno je implementirati rukovatelje događajima, svi rukovatelji događajima uzimaju argument izveden iz wxEventa čija se točna klasa razlikuje prema vrsti događaja i klasi izvornog prozora. Za implementaciju ove klase koriste se wxCommandEvent-i koji su vezani za naredbe izbornika i većinu kontrolnih naredbi (kao što su pritisci na gumb).

Primjer kako pozvati rukovatelj događaja kao argument metodi:

```
void MyFrame::OnExit(wxCommandEvent& event) {
      //zatvori prozor i oslobodi sve korištene resurse
      this->Destroy();
}
```

Kôd 6: primjer pozivanja rukovoditelja događaja kao argument metodi

Da bi se koristila tablica događaja, kao što je prethodno objašnjeno prvo je potrebno odlučiti u kojoj klasi se žele obraditi događaji. U ovom slučaju događaji se obrađuju u klasi imena Notepad. Nije važno gdje se definicija obrade događaja u klasi nalazi, no uobičajeno ju je staviti na kraj jer makronaredba interno mijenja vrstu pristupa, pa je najsigurnija, ako ništa ne slijedi. Primjer sintakse je prikazan u poglavlju 5.2.2

U kôdu aplikacije tablica događaja je podijeljena u sedam grupa. Prva grupa događaja odnosi se na karticu *File*. Kartica *File* sadrži sljedeće pozive na događaje:

```
//Kartica file:
EVT_MENU(idFileNew, Notepad::OnFileNew)
EVT_MENU(idFileNewWindow, Notepad::OnFileNewWindow)
EVT_MENU(idFileOpen, Notepad::OnFileOpen)
EVT_MENU(idFileSave, Notepad::OnFileSave)
EVT_MENU(idFileSaveAs, Notepad::OnFileSaveAs)
EVT_MENU(idFilePageSetup, Notepad::OnFilePageSetup)
EVT_MENU(idFilePrint, Notepad::OnFilePrint)
EVT_MENU(idFileExit, Notepad::OnFileExit)
```

Kôd 7: događaji vezani za karticu *File*

Druga grupa događaja odnosi se na karticu *Edit.* Kartica *Edit* sadrži sljedeće pozive na događaje:

```
//kartica edit:
EVT_MENU(idEditUndo, Notepad::OnEditUndo)
EVT_MENU(idEditCut, Notepad::OnEditCut)
EVT_MENU(idEditCopy, Notepad::OnEditCopy)
EVT_MENU(idEditPaste, Notepad::OnEditPaste)
EVT_MENU(idEditDelete, Notepad::OnEditDelete)
EVT_MENU(idEditFind, Notepad::OnEditFind)
EVT_MENU(idEditReplace, Notepad::OnEditReplace)
EVT_MENU(idEditGoto, Notepad::OnEditGoto)
EVT_MENU(idEditSelectAll, Notepad::OnEditSelectAll)
EVT_MENU(idEditTimeDate, Notepad::OnEditTimeDate)
```

Kôd 8: događaji vezani za karticu *Edit*

Treća grupa događaja odnosi se na karticu *Format.* Kartica *Format* sadrži sljedeće pozive za događaje:

```
//kartica format:
EVT_MENU(idFormatAutoLine, Notepad::OnFormatAutoLine)
```

```
EVT_MENU(idFormatWordWrap, Notepad::OnFormatWordWrap)
EVT_MENU(idFormatFont, Notepad::OnFormatFont)
```

Kôd 9: događaji vezani za karticu *Format*

Četvrta grupa događaja odnosi se na karticu *View*. Kartica *View* sadrži sljedeće pozive na događaje:

```
//kartica view:
EVT_MENU(idViewStatBar, Notepad::OnViewStatBar)
```

Kôd 10: događaji vezani za karticu *View*

Peta grupa događaja odnosi se na karticu *About*. Kartica *About* sadrži sljedeće pozive na događaje:

```
//kartica about:
EVT_MENU(idHelpAbout, Notepad::OnHelpAbout)
```

Kôd 11: događaji vezani za karticu *About*

Šesta grupa događaja odnosi se na događaj kada je pritisnut gumba X:

```
//event na klik X gumba okvira:
EVT_CLOSE(Notepad::OnXButton)
```

Kôd 12: događaj koji se odnosi na pritisak gumba X

Sedma grupa događaja odnosi se na događaje ažuriranja korisničkog sučelja, aplikacija se mora potruditi provjeriti/poništiti, omogućiti/onemogućiti, prikazati/sakriti i postaviti tekst za elemente kao što su stavke izbornika i gumbi na alatnoj traci. Kôd za to mora biti pomiješan s kodom koji se poziva kada se pokrene radnja za stavku izbornika ili gumb. Odnosno, događaje ažuriranja korisničkog sučelja definirana rukovoditeljima događaja koji gledaju stanje aplikacije i u skladu s tim mijenjaju elemente korisničkog sučelja. Ovo je preporučano staviti u aplikacije koje koriste klasu wxTextCtrl.

```
//wxUpdateUIEvent:
EVT_UPDATE_UI(idEditUndo, Notepad::OnUpdateUIUndo)
EVT_UPDATE_UI(idEditCut, Notepad::OnUpdateUICut)
EVT_UPDATE_UI(idEditCopy, Notepad::OnUpdateUICopy)
EVT_UPDATE_UI(idEditPaste, Notepad::OnUpdateUIPaste)
```

Kôd 13: prikaz sedme grupe događaja

5.2.5. Implementacija aplikacije – “main“ funkcija

Kao i u svim programima, mora postojati glavna funkcija (eng. main function). Kod wxWidgets-a main funkcija implementira se pomoću sljedeća makronaredbe, koja stvara instancu aplikacije i pokreće program.

```
wxIMPLEMENT_APP(MainApp)
```

Kôd 14: prikaz sintakse implementacije main funkcije u wxWidgets

Kao što je prethodno spomenuto u poglavlju 5.2.2 klasa se poziva pri pokretanju programa. Trebalo bi se koristiti za inicijalizaciju programa, kao npr. za kreiranje glavnog prozora programa ili nekoliko njih. U ovom slučaju razvoja aplikacije kreira se glavni prozor OIRI Notepad tekstualnog editora.

5.3. UML dijagram

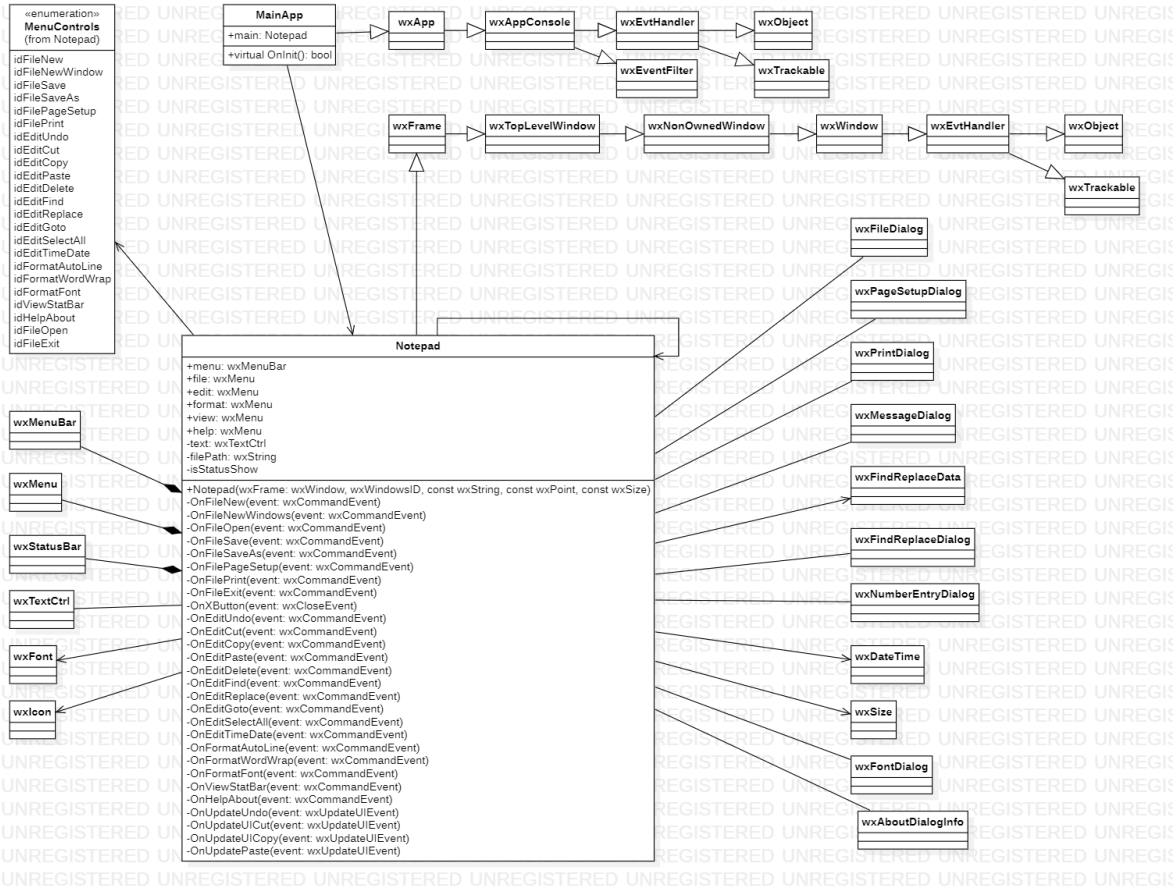
UML (eng. *Unified Modeling Language*) je jezik za specifikaciju, vizualizaciju, konstruiranje i dokumentiranje programskog sustava, poslovnog sustava i drugih ne programskih sustava.

Može se koristiti za poslovno modeliranje, softversko modeliranje u svim fazama razvoja i za sve tipove sustava, te opće modeliranje u svim fazama razvoja i za sve tipove sustava, te opće modeliranje bilo koje konstrukcije koja ima statičku strukturu i dinamičko ponašanje.

5.3.1. Klasni dijagram

Pokazuje statičku strukturu sustava definirajući elemente klasifikatora (klase, sučelja, paketi, veze, instance, …) i statične odnose među njima. Na dijagramu klasa prikazana je unutarnja struktura klasa koju distribuiraju atributi i operacije. Sustav najčešće ima brojne dijagrame klasa jer nisu sve klase uključene u jedan dijagram klase. Isto tako neka klasa može, kada je potrebno, sudjelovati u nekoliko dijagrama klasa.

U sljedećem dijelu ovog poglavlja bit će prikazan klasni dijagram koji odgovara razvoju OIRI Notepad aplikacije.



Slika 9: prikaz klasnog dijagrama OIRI Notepad aplikacije

U dijagramu klasa, vidljivom na Slika 9 prikazane su klase korištene u izradi OIRI Notepad aplikacije te njihovi međusobni odnosi.

6. Upute za build programa

U ovom poglavlju bit će prikazano kako *build*-ati aplikaciju na svakoj od platformi (Windows, Linux, Mac OS) koristeći jedan kôd skripte.

Bitno je napomenuti da je kôd skripte za *build* aplikacije koristeći wxWidgets preuzet od „Just Dev Tutorials“. (Just Dev Tutorials, 2022)

Cilj autora od kojeg je preuzet kôd skripte za *build* aplikacije koristeći wxWidgets je:

- Koristeći isti skup naredbi aplikacija se može izgraditi na Windows, Linux i Mac OS platformi
- Skripta za *build* automatski detektira okolinu, kompajler itd. za korištenje i sastavljanje aplikacije
- Skripta provjerava da li je biblioteka (wxWidgets u ovom slučaju) već instalirana u operacijskom sustavu. Ako se biblioteka ne pronađe, skripta tada automatski preuzima, prevodi, i instalira biblioteku u privremeni direktorij

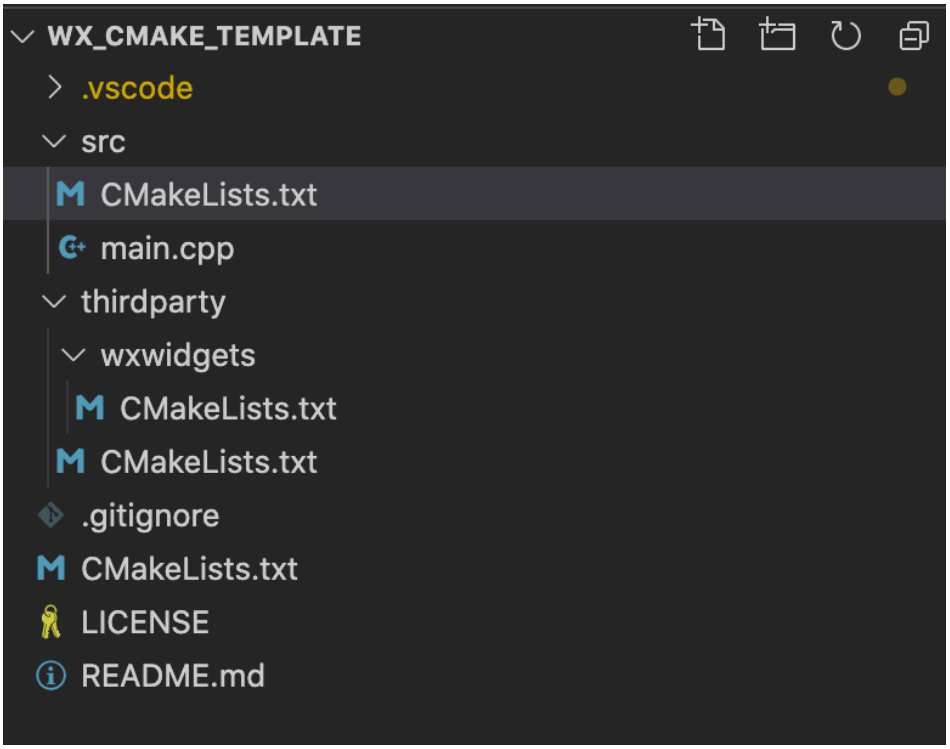
Kako autor spominje u tekstu, na ovaj način se pojednostavljuje proces izgradnje za svaku platformu. Programerima se na ovaj način uvelike pojednostavljuje posao.

6.1. Korišteni alati i biblioteke

Za *build* same aplikacije korištena je već prethodno spomenuta biblioteka wxWidget i novo spomenuta biblioteka CMake. Cmake je generator sustava izgradnje koji je u suštini standard za C++ razvoj. Iako, sintaksa CMake može na prvu izgledati zastarjelo, a sam sustav je poprilično kompliciran, noviji dodaci za CMake „*ExternalProject*“ i „*FetchContent*“ funkcionalnosti daju moderniji sustav izgradnje aplikacija i učinkovito moderno upravljanje ovisnostima.

6.2. Kôd skripte

Cijeli izvor oglednog kôda može se preuzeti s github-a na sljedećoj web poveznici: https://github.com/lszl84/wx_cmake_template.



Slika 10: prikaz strukture kôda skripte

Datoteke CMakeLists.txt su skripte za izgradnju koje koriste CMake sustav. Kao što je vidljivo sa Slika 10 pronalazimo tri datoteke CMakeList.txt i svaka se nalazi u odvojenom direktoriju.

- CMakeList.txt u korijenskom direktoriju je glavna skripta za izgradnju i uključuje sve ostale skripte
- CMakeList.txt u *src* direktoriju je skripta za izgradnju glavnog projekta
- CMakeList.txt u *thirdparty* direktoriju je skripta za pronalazak ili preuzimanje biblioteke wxWidgets

Detalji kôda svake skripte u ovom poglavlju neće biti objašnjeni jer je cilj objasnit proces *build*-a aplikacije.

6.3. Build proces aplikacije

Prije build-a projekta, potrebno se je uvjeriti da alati i biblioteke (wxWidgets i CMake) instalirani na sustav.

6.3.1. Build aplikacije na Windows platformi

Prvo je potrebno instalirati Visual Studio i prilikom instalacije potrebno je instalirati komponentu imena „Desktop development with C++“. Također, potrebno je instalirati *Git* softver koristeći sljedeću naredbu, naredba se pokreće u Windows naredbenom retku (cmd):

```
winget install --id Git.Git -e --source winget
```

Naredba 1: naredba za instalaciju Git softvera

Kad su navedene dvije ovisnosti instalirane potrebno je pokrenuti naredbeni redak Visual Studia. Puni naziv naredbenog retka kojeg je potrebno otvoriti je: „Native Tools Command Prompt for VS 2019“. Kad je otvoren navedeni naredbeni redak potrebno je slijediti sljedeće naredbe:

Kreirati željeni direktorij i pozicionirati se u njega za preuzimanje kôda navedene skripte sa *Github*-a:

```
Cd /d D:\
Mkdir ajanach_projekt
Cd ajanach_projekt
git clone https://github.com/lszl84/wx_cmake_template.git
```

Naredba 2: kreiranje direktorija i preuzimanja kôda skripte za *build* na Windows platformi

Pozicionirati se u preuzeti direktorij sa *Git* naredbom i pokrenuti CMake naredbu za konfiguraciju projekta:

```
Cd wx_make_template
cmake -S. -Bbuild
```

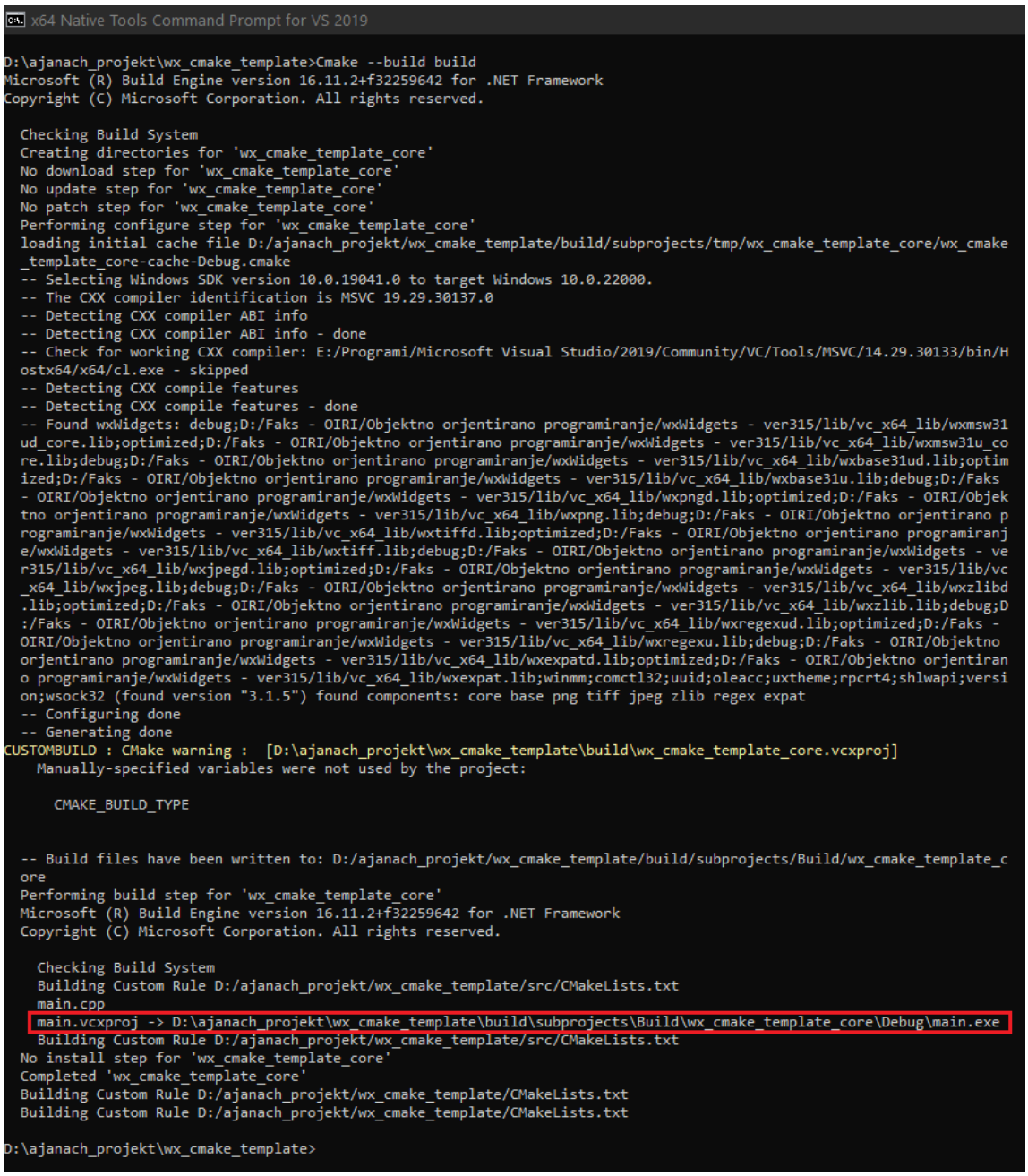
Naredba 3: Cmake naredba za konfiguraciju projekta za *build* na Windows platformi

Na kraju pokrenuti CMake naredbu za build projekta:

```
Cmake --build build
```

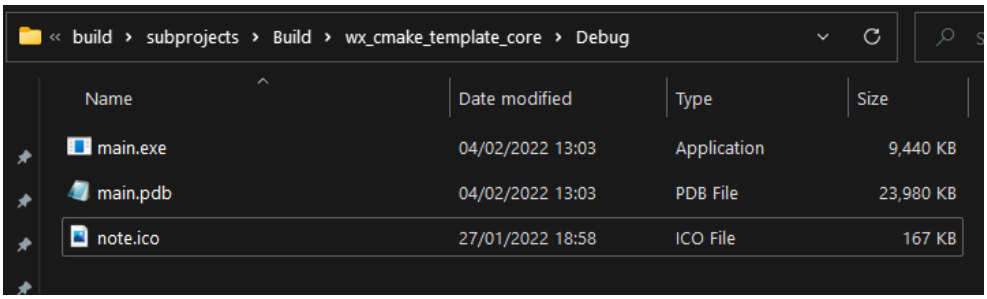
Naredba 4: CMake naredba za *build* projekta na Windows platformi

Naredbi za *build* aplikacije treba neko vrijeme jer treba preuzet i instalirati wxWidgets biblioteku (ako je nema), i prevest C++ kôd same aplikacije.



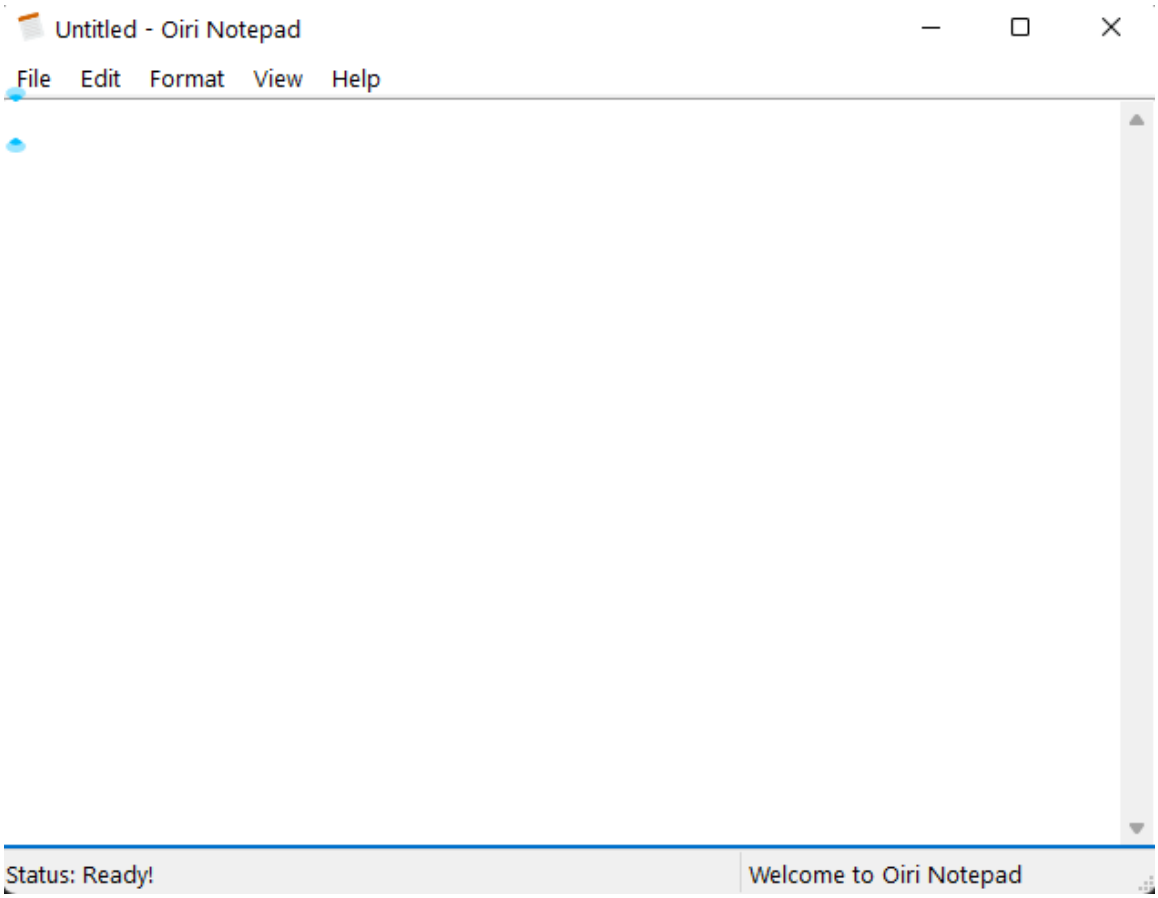
Slika 11: prikaz izlaza zadnje pokrenute naredbe

Kao što je vidljivo iz Slika 11 build aplikacije nalazi na navedenoj putanji.



Slika 12: prikaz sadržaja putanje gdje se nalazi *build* aplikacije

Da bi build aplikacije normalno radio potrebno je za kraj dodati ovisnost od koje aplikacija ovisi, a to je ikona. Ovdje završava poglavlje za *build* aplikacije.



Slika 13: prikaz sučelja OIRI Notepad aplikacije

Kao što je vidljivo na Slika 13 aplikacija je pokrenuta koristeći main.exe datoteku.

6.3.2. Build aplikacije na Mac OS platformi

Prvo je potrebno instalirati „*build tools*“ koristeći *homebrew* (besplatan paketni upravitelj za Mac OS) tako da se otvori naredbeni i upiše sljedeća naredba:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/instal
l.sh)"
```

Naredba 5: prikaz instalacije „*Build tools*“ na Mac OS platformi

Instalirati CMake alat:

```
Brew install cmake
```

Naredba 6: instalacija cmake alata za Mac OS platformu

Preuzeti kôd skripte u željeni direktorij:

```
git clone https://github.com/lsz184/wx_cmake_template.git
```

Naredba 7: preuzimanje kôda skripte sa git huba za Mac OS platformu

Pozicionirati se u preuzeti direktorij sa git naredbom i pokrenuti CMake naredbu za konfiguraciju projekta:

```
Cd wx_make_template
cmake -S. -Bbuild
```

Naredba 8: Cmake naredba za konfiguraciju projekta za *build* na Mac OS platformi

Na kraju pokrenuti CMake naredbu za build projekta:

```
Cmake --build build
```

Naredba 9: CMake naredba za *build* projekta na Mac OS platformi

6.3.3. Build aplikacije na Linux platformi

Prvo je potrebno instalirati „Build system“ i „UI libraries“ za Linux u ovom slučaju GTK tako da se otvori naredbeni redak (terminal):

```
Sudo apt-get install libgtk-3-dev -y
```

Naredba 10: prikaz instalacije ovisnosti za Linux platformu

Instalirati Git softver i preuzeti kôd skripte u željeni direktorij:

```
Sudo apt-get install git -y
git clone https://github.com/lszl84/wx_cmake_template.git
```

Naredba 11: preuzimanje kôda skripte sa git huba za Linux platformu

Pozicionirati se u preuzeti direktorij sa git naredbom i pokrenuti CMake naredbu za konfiguraciju projekta:

```
Cd wx_make_template
cmake -S. -Bbuild
```

Naredba 12: Cmake naredba za konfiguraciju projekta za *build* na Linux platformi

Na kraju pokrenuti CMake naredbu za build projekta:

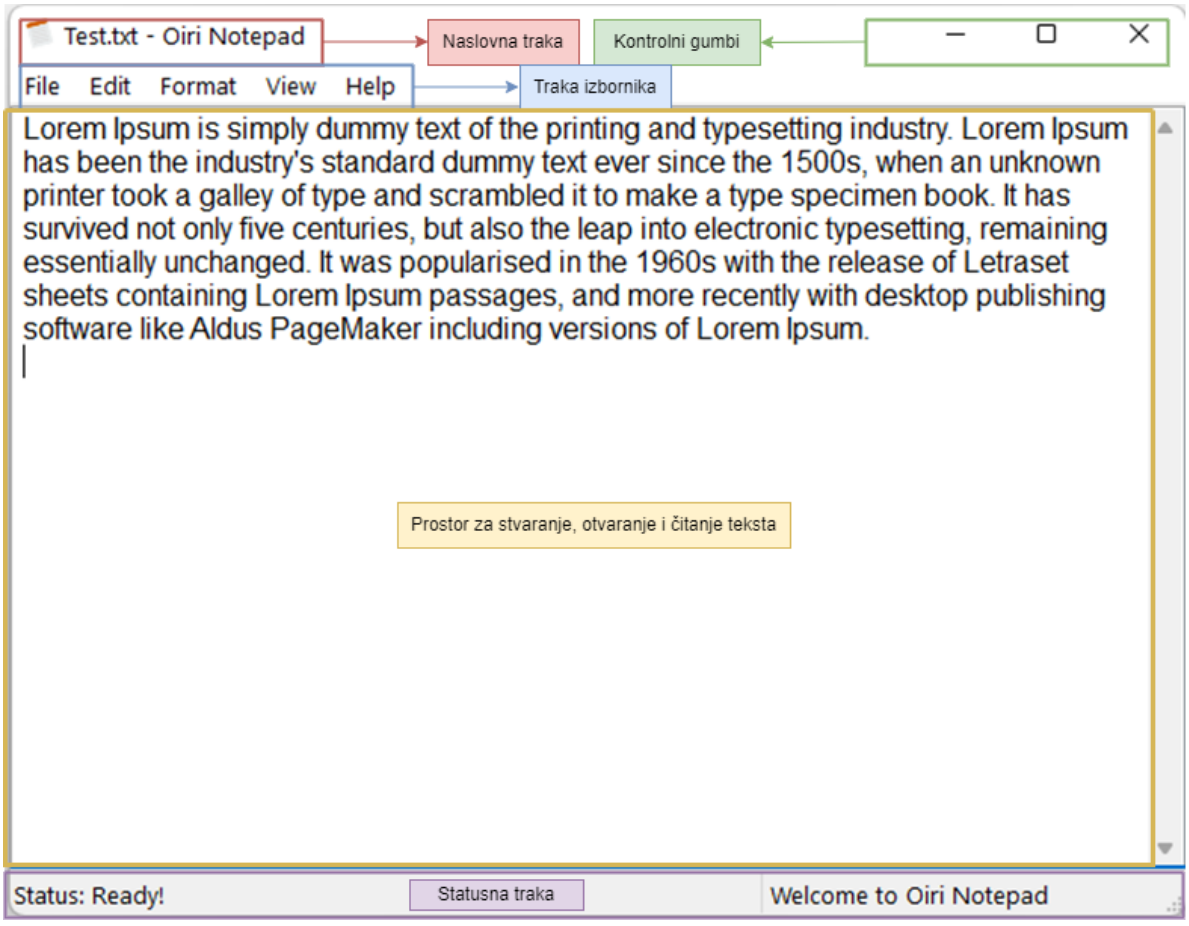
```
Cmake --build build
```

Naredba 13: CMake naredba za *build* projekta na Linux platformi

7. Osnovne upute za korištenje aplikacije

U ovom poglavlju bit će prikazan izgled prozora OIRI Notepad aplikacije za uređivanje teksta. Također, biti te dane osnovne upute za korištenje aplikacije.

Nakon pokretanja OIRI Notepad aplikacije, pojavljuje se prozor koji se sastoji od sljedećih elemenata: **naslovna traka –** nalazi se na vrhu prozora, na njoj piše ime trenutno aktivnog dokumenta i ime aplikacije. U desnom uglu naslovne trake nalaze se **kontrolni gumbi**, odnosno gumb za minimiziranje prozora programa, smanjenje/maksimiziranje prozora aplikacije i gumb za zatvaranje aplikacije. Ispod naslovne trake nalazi se traka s tekstualnim izbornicima **File, Edit, Format** itd. Klikom na ime izbornika pojavljuje se **padajući izbornik s** naredbama koje su grupirane prema namjeni. Tako izbornik „File“ sadrži naredbe za rad s datotekama, izbornik „Edit“ sadrži naredbe za uređivanje dokumenta itd. U središtu okvira nalazi se prostor za stvaranje, otvaranje i čitanje teksta. Na dnu okvira nalazi se **statusna traka**.



Slika 14: izgled korisničkog sučelja OIRI Notepad aplikacije

Što se tiče korisničkih uputa mislim da je aplikacije poprilično jednostavna za korištenje. Korisnik unosi tekst u prostor za stvaranje, otvaranje i čitanje teksta. Zatim, korisnik koristi naredbe iz tekstualnog izbornika kako bi manipulirao tekstom i tekstualnom datotekom. Svaka naredba iz izbornika detaljno je opisana u poglavlju 5.2.2. Klasa Notepad.

Zaključak

Tijekom izrade projekta, stavljen je naglasak na dvije stvari: izrada tehničke dokumentacije korištenjem wxWidgets biblioteke, i izrada pristupačnog, jednostavnog i funkcionalnog sučelja za stvaranje, otvaranje i čitanje tekstualnih datoteka.

Biblioteka *wx* se pokazala izvrsnom za svaki dio problematike rada vezan uz prikaz i manipulaciju prikazanim objektima. Kroz cijeli rad se pokazalo da se biblioteka može koristiti za izradu aplikacije koja može izgledati kao bilo koja komercijalna. Jedini problem vezan uz tu biblioteku je to što je dokumentacija opsežna, a svejedno ponekad malo toka kaže.

Budući da je danas računalo neophodno u gotovo svim područjima života. Sve se više razvija elektroničko poslovanje i upotreba ICT tehnologije. Time se smanjuju mnogi troškovi kojima su izloženi kupci i proizvođači. Većim razvojem dolazi i do veće potrebe za programerima te razvojem aplikacija i programa.

Programski jezik C++ je kompleksan i nalazi se na ljestvici najpopularnijih programskih jezika današnjice. Mnoge su se aplikacije i programi razvili zahvaljujući njemu. Kod korištenja ovog programskoj jezika, kao i kod ostalih, potrebno se pridržavati određenih pravila sintakse i razmišljati na logičan način.

Uz programski jezik C++ postoji još mnogo kvalitetnih i dobrih programskih jezika. Stoga, ako se razumije jedan programski jezik, bit će vrlo jednostavno raditi i u drugim programskim okruženjima.

Popis slika

Popis kôdova

Popis naredba

Literatura

Svaki autor piše popis literature na kraju rada. Popis literature se piše stilom literatura.

[1] B. HOOKWAY, „Chapter 1: The Subject of the Interface“, B. Hookway Interface, pp. 1-58, 2014.

[2] W3SCHOOLS, What is CLI, https://www.w3schools.com/whatis/whatis_cli.asp, veljača 2022.

[3] STÉPHANE RICHARD, Text User Interface Development Series: Part One – T.U.I. Basics, http://www.petesqbsite.com/sections/express/issue21/tuiseriespart1.htm, veljača 2022.

[4] W. SHOTTS, The Linux Command Line, 2019.

[5] A. POWELL, Web 101: A History of the GUI, Wired, https://www.wired.com/1997/12/web-101-a-history-of-the-gui/ veljača 2022.

[6] Apple Lisa Personal Computer

[7] Screenshot of Windows 95, 2017.

[8] Microsoft, Windows 10

[9] W3SCHOOLS, What is C++, https://www.w3schools.com/cpp/cpp_intro.asp, veljača 2022.

[10] TUTORIALSPOINT, C++ Inheritance, https://www.tutorialspoint.com/cplusplus/cpp_inheritance.htm, veljača 2022.

[11] Microsoft Visual studio, Wikipedia, https://en.wikipedia.org/wiki/Microsoft_Visual_Studio, veljača 2022.

[12] Just Dev Tutorials, wxWidgets + CMake: Multiplatform Superbuild, https://justdevtutorials.medium.com/wxwidgets-cmake-multiplatform-superbuild-4ea86c4e6eda, veljača 2022.

Prilog: programski kod

```
//uključena zaglavlja:
#include <wx/wx.h>
#include <wx/richmsgdlg.h>
#include <wx/printdlg.h>
#include <wx/string.h>
#include <wx/fdrepdlg.h>
#include <wx/numdlg.h>
#include <wx/fontdlg.h>
#include <wx/aboutdlg.h>
#include <wx/msgdlg.h>
#include <string>

class Notepad : public wxFrame {
public:
    Notepad() : wxFrame(NULL, wxID_ANY, wxT("Oiri Notepad"), wxDefaultPosition,
 wxSize(650, 500)) {
        wxMenuBar* menu = new wxMenuBar();
        wxMenu* file = new wxMenu();
        wxMenu* edit = new wxMenu();
        wxMenu* format = new wxMenu();
        wxMenu* view = new wxMenu();
        wxMenu* help = new wxMenu();

        menu->Append(file, wxT("&File"));
        menu->Append(edit, wxT("&Edit"));
        menu->Append(format, wxT("&Format"));
        menu->Append(view, wxT("&View"));
        menu->Append(help, wxT("&Help"));

        file->Append(idFileNew, wxT("&New File\tCtrl+N"));
        file->Append(idFileNewWindow, wxT("&New Window\tCtrl+Shift+N"));
        file->Append(idFileOpen, wxT("&Open File\tCtrl+O"));
        file->Append(idFileSave, wxT("&Save\tCtrl+S"));
        file->Append(idFileSaveAs, wxT("&Save As...\tCtrl+Shift+S"));
        file->AppendSeparator();
        file->Append(idFilePageSetup, wxT("&Page Setup"));
        file->Append(idFilePrint, wxT("&Print\tCtrl+P"));
        file->AppendSeparator();
        file->Append(idFileExit, wxT("&Exit\tCtrl+W"));

        edit->Append(idEditUndo, wxT("&Undo\tCtrl+Z"));
        edit->AppendSeparator();
        edit->Append(idEditCut, wxT("&Cut\tCtrl+X"));
        edit->Append(idEditCopy, wxT("&Copy\tCtrl+C"));
        edit->Append(idEditPaste, wxT("&Paste\tCtrl+V"));
        edit->Append(idEditDelete, wxT("&Delete\tDel"));
        edit->AppendSeparator();
        edit->Append(idEditFind, wxT("&Find\tCtrl+F"));
        edit->Append(idEditReplace, wxT("&Replace\tCtrl+H"));
        edit->Append(idEditGoto, wxT("&Goto\tCtrl+G"));
        edit->AppendSeparator();
        edit->Append(idEditSelectAll, wxT("&Select All\tCtrl+A"));
        edit->Append(idEditTimeDate, wxT("&Time Date\tF5"));
```

```
        format->Append(idFormatAutoLine, "&Auto Line");
        format->Append(idFormatWordWrap, wxT("&Word wrap"));
        format->Append(idFormatFont, wxT("&Font..."));

        view->Append(idViewStatBar, wxT("&Status bar"));

        help->Append(idHelpAbout, wxT("&About"));

        text = new wxTextCtrl(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDef
aultSize, wxTE_PROCESS_ENTER | wxTE_MULTILINE);
        this->text-
>SetFont(wxFont(11, wxFONTFAMILY_DEFAULT, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORM
AL, false, wxEmptyString));
        this->text->SetForegroundColour(*wxBLACK);
        this->text->SetBackgroundColour(*wxWHITE);

        this-
>SetIcon(wxIcon(wxT("note.ico"), wxBITMAP_TYPE_ICO)); //postavljanje ikone okvi
ra
        this-
>SetLabel("Untitled - Oiri Notepad"); //Postavljanje naslova okvira
        this->SetMenuBar(menu);

        this->filePath = ""; //dodijeljena vrijednost varijabli
        this->isStatusShow = false;

        //kreiranje status bar-a:
        const int size = 2;
        this->CreateStatusBar(2, wxSTB_DEFAULT_STYLE | wxSB_SUNKEN);
        this->SetStatusText(wxT("Status: Ready!"), 0);
        this->SetStatusText(wxT("Welcome to Oiri Notepad"), 1);
        int a[size] = { -1, 200 };
        this->GetStatusBar()->SetStatusWidths(size, a);
        this->GetStatusBar()->Show();
    }

private:
    wxTextCtrl* text;
    wxString filePath;
    bool isStatusShow;

    void OnFileNew(wxCommandEvent& event) {
        text->SetValue("");
        SetLabel("Untitled - Oiri Notepad");
        filePath = "";

        //status:
        PushStatusText(wxT("Status: New file"));
        wxSleep(1.5);
        PopStatusText();
    }

    void OnFileNewWindow(wxCommandEvent& event) {
        Notepad* newWindow = new Notepad();
        newWindow->Show();

        //status:
```

```
        PushStatusText(wxT("Status: Opening new window"));
        wxSleep(1.5);
        PopStatusText();
    }

    void OnFileOpen(wxCommandEvent& event) {
        wxFileDialog* openDialog = new wxFileDialog(this, wxT("Open"), wxT(""),
 wxT(""), wxT("Text Files (*.txt)|*.txt|All Files (*.*)|*.*|C++ Files (*.cpp)|*
.cpp"), wxFD_OPEN);
        if (openDialog->ShowModal() == wxID_OK) {
            this->text->LoadFile(openDialog->GetPath());
            filePath = openDialog->GetPath();
            SetLabel(openDialog->GetFilename() + " - Oiri Notepad");

            //status:
            PushStatusText(wxT("Status: ") + openDialog-
>GetFilename() + wxT(" is opened"));
            wxSleep(2);
            PopStatusText();
        }
    }

    void OnFileSave(wxCommandEvent& event) {
        if (filePath != "") {
            text->SaveFile(filePath);

            //status:
            PushStatusText(wxT("Status: File is saved"));
            wxSleep(1.5);
            PopStatusText();

            return;
        }

        wxFileDialog* saveDialog = new wxFileDialog(this, wxT("Save As"), wxEmp
tyString, wxT(""), wxT("Text Files (*.txt)|*.txt|All Files (*.*)|*.*|C++ Files
(*.cpp)|*.cpp"), wxFD_SAVE | wxFD_OVERWRITE_PROMPT);
        saveDialog->SetFilterIndex(1);
        if (saveDialog->ShowModal() == wxID_OK) {
            text->SaveFile(saveDialog->GetPath().c_str());
            filePath = saveDialog->GetPath();
            SetLabel(saveDialog->GetFilename() + " - Oiri Notepad");

            //status:
            PushStatusText(wxT("Status: File is saved as ") + saveDialog-
>GetFilename());
            wxSleep(2);
            PopStatusText();
        }
    }

    void OnFileSaveAs(wxCommandEvent& event) {
        wxFileDialog* saveAsDialog = new wxFileDialog(this, wxT("Save File~"),
wxT(""), wxT(""), wxT("Text Files (*.txt)|*.txt|All Files (*.*)|*.*|C++ Files (
*.cpp)|*.cpp"), wxFD_SAVE);
        saveAsDialog->SetFilterIndex(1);
        if (saveAsDialog->ShowModal() == wxID_OK) {
            text->SaveFile(saveAsDialog->GetPath().c_str());
```

```
            filePath = saveAsDialog->GetPath();
            SetLabel(saveAsDialog->GetFilename() + " - Oiri Notepad");

            //status:
            PushStatusText(wxT("Status: File is saved as ") + saveAsDialog-
>GetFilename());
            wxSleep(2);
            PopStatusText();
        }
    }

    void OnFilePageSetup(wxCommandEvent& event) {
        PushStatusText(wxT("Status: Page setup"));
        wxSleep(1.5);
        PopStatusText();
        wxPageSetupDialog* pageSetupDialog = new wxPageSetupDialog(this);
        pageSetupDialog->ShowModal();

        //status:
        PushStatusText(wxT("Status: Page setup configuration is saved."));
        wxSleep(1.5);
        PopStatusText();
    }

    void OnFilePrint(wxCommandEvent& event) {
        PushStatusText(wxT("Status: Print"));
        wxSleep(1.5);
        PopStatusText();
        wxPrintDialog* printDialog = new wxPrintDialog(this);
        printDialog->ShowModal();

        //status:
        PushStatusText(wxT("Status: Printing..."));
        wxSleep(1.5);
        PopStatusText();
    }

    void OnFileExit(wxCommandEvent& event) {
        if (text->GetValue() == "")
        {
            //status:
            PushStatusText(wxT("Status: Exiting..."));
            wxSleep(1.5);

            this->Destroy();
        }

        else
        {
            wxMessageDialog* messageDialog = new wxMessageDialog(this, wxT("Do
you want to save changes?"), wxT("Info"), wxYES_NO | wxCANCEL | wxYES_DEFAULT |
 wxICON_QUESTION);

            switch (messageDialog->ShowModal())
            {
            case wxID_YES:
                if (filePath != "") {
                    text->SaveFile(filePath);
```

```
                    //status:
                    PushStatusText(wxT("Status: File is saved"));
                    wxSleep(1.5);
                    PushStatusText(wxT("Status: Exiting..."));
                    wxSleep(1.5);

                    this->Destroy();
                    return;
                }

                if (filePath == "")
                {
                    wxFileDialog* saveDialog = new wxFileDialog(this, wxT("Save
 As"), wxEmptyString, wxT(""), wxT("Text Files (*.txt)|*.txt|All Files (*.*)|*.
*|C++ Files (*.cpp)|*.cpp"), wxFD_SAVE | wxFD_OVERWRITE_PROMPT);
                    saveDialog->SetFilterIndex(1);
                    if (saveDialog->ShowModal() == wxID_OK) {
                        text->SaveFile(saveDialog->GetPath().c_str());
                        filePath = saveDialog->GetPath();
                        SetLabel(saveDialog-
>GetFilename() + " - Oiri Notepad");

                        //status:
                        PushStatusText(wxT("Status: File is saved as ") + saveD
ialog->GetFilename());
                        wxSleep(2);
                        PushStatusText(wxT("Status: Exiting..."));
                        wxSleep(1.5);

                        this->Destroy();
                    }
                }
                break;
            case wxID_NO:
                //status:
                PushStatusText(wxT("Status: Exiting..."));
                wxSleep(1.5);

                this->Destroy();
                break;
            case wxID_CANCEL:
                PushStatusText(wxT("Status: Action is canceled"));
                wxSleep(1.5);
                PopStatusText();

                break;
            default:
                PushStatusText(wxT("Status: Error"));
                wxSleep(1.5);
                PopStatusText();
                break;
            }
        }
    }

    void OnXButton(wxCloseEvent& event) {
        if (text->GetValue() == "")
```

```
        {
            //status:
            PushStatusText(wxT("Status: Exiting..."));
            wxSleep(1.5);

            this->Destroy();
        }

        else
        {
            wxMessageDialog* messageDialog = new wxMessageDialog(this, wxT("Do
you want to save changes?"), wxT("Info"), wxYES_NO | wxCANCEL | wxYES_DEFAULT |
 wxICON_QUESTION);

            switch (messageDialog->ShowModal())
            {
            case wxID_YES:
                if (filePath != "") {
                    text->SaveFile(filePath);

                    //status:
                    PushStatusText(wxT("Status: File is saved"));
                    wxSleep(1.5);
                    PushStatusText(wxT("Status: Exiting..."));
                    wxSleep(1.5);

                    this->Destroy();
                    return;
                }

                if (filePath == "")
                {
                    wxFileDialog* saveDialog = new wxFileDialog(this, wxT("Save
 As"), wxEmptyString, wxT(""), wxT("Text Files (*.txt)|*.txt|All Files (*.*)|*.
*|C++ Files (*.cpp)|*.cpp"), wxFD_SAVE | wxFD_OVERWRITE_PROMPT);
                    saveDialog->SetFilterIndex(1);
                    if (saveDialog->ShowModal() == wxID_OK) {
                        text->SaveFile(saveDialog->GetPath().c_str());
                        filePath = saveDialog->GetPath();
                        SetLabel(saveDialog-
>GetFilename() + " - Oiri Notepad");

                        //status:
                        PushStatusText(wxT("Status: File is saved as ") + saveD
ialog->GetFilename());
                        wxSleep(2);
                        PushStatusText(wxT("Status: Exiting..."));
                        wxSleep(1.5);

                        this->Destroy();
                    }
                }
                break;
            case wxID_NO:
                //status:
                PushStatusText(wxT("Status: Exiting..."));
                wxSleep(1.5);
```

```
            this->Destroy();
            break;
        case wxID_CANCEL:
            PushStatusText(wxT("Status: Action is canceled"));
            wxSleep(1.5);
            PopStatusText();

            break;
        default:
            PushStatusText(wxT("Status: Error"));
            wxSleep(1.5);
            PopStatusText();
            break;
        }
    }
}

void OnEditUndo(wxCommandEvent& event) {
    //status:
    PushStatusText(wxT("Status: undo"));
    wxSleep(1);
    PopStatusText();

    text->Undo();
}

void OnEditCut(wxCommandEvent& event) {
    //status:
    PushStatusText(wxT("Status: Cut"));
    wxSleep(1);
    PopStatusText();

    text->Cut();
}

void OnEditCopy(wxCommandEvent& event) {
    //status:
    PushStatusText(wxT("Status: Copy"));
    wxSleep(1);
    PopStatusText();

    text->Copy();
}

void OnEditPaste(wxCommandEvent& event) {
    //status:
    PushStatusText(wxT("Status: Paste"));
    wxSleep(1);
    PopStatusText();

    text->Paste();
}

void OnEditDelete(wxCommandEvent& event) {
    //status:
    PushStatusText(wxT("Status: Delete"));
    wxSleep(1);
    PopStatusText();
```

```
        text->RemoveSelection();
    }

    void OnEditFind(wxCommandEvent& event) {
            wxFindReplaceData* findData = new wxFindReplaceData();
            PushStatusText(wxT("Status: Find"));
            wxSleep(1.5);
            PopStatusText();
            wxFindReplaceDialog* find = new wxFindReplaceDialog(this, findData,
 wxT("Find dialog"), wxFR_MATCHCASE);
            find->Show(true);

            //status:
            PushStatusText(wxT("Status: Finding words"));
            wxSleep(1.5);
            PopStatusText();
    }

    void OnEditReplace(wxCommandEvent& event) {
        wxFindReplaceData* findData = new wxFindReplaceData();
        PushStatusText(wxT("Status: Replace"));
        wxSleep(1.5);
        PopStatusText();
        wxFindReplaceDialog* replace = new wxFindReplaceDialog(this, findData,
wxT("Find and Replace Dialog"), wxFR_REPLACEDIALOG | wxFR_NOWHOLEWORD);
        replace->Show(true);

        //status:
        PushStatusText(wxT("Status: Replacing words"));
        wxSleep(1.5);
        PopStatusText();
    }

    void OnEditGoto(wxCommandEvent& event) {
        PushStatusText(wxT("Status: Go to"));
        wxSleep(1.5);
        PopStatusText();
        long lineNumber = wxGetNumberFromUser(wxT(""), wxT(""), wxT("Go to the
following line"), 1, 0, 100000, this); //moze se koristiti i int, ali XYToPosit
ion zahtijeva long tip podatka
        if (lineNumber >= 0)
        {
            text->SetInsertionPoint(text->XYToPosition(0, lineNumber));

            //status:
            PushStatusText(wxT("Status: Go to line number: ") + std::to_string(
lineNumber));
            wxSleep(1.5);
            PopStatusText();
        }
    }

    void OnEditSelectAll(wxCommandEvent& event) {

        //status:
        PushStatusText(wxT("Status: Select all"));
        wxSleep(1.5);
```

```
        PopStatusText();
        text->SelectAll();
    }

    void OnEditTimeDate(wxCommandEvent& event) {
        long from, to;
        text->GetSelection(&from, &to);
        wxDateTime* dateAndTime = new wxDateTime();
        dateAndTime->SetToCurrent();
        text->Replace(from, to, dateAndTime->Format());

        //status:
        PushStatusText(wxT("Status: Current time and date: " + dateAndTime-
>Format()));
        wxSleep(1.5);
        PopStatusText();
    }

    void OnFormatAutoLine(wxCommandEvent& event) {
        long flag = text->GetWindowStyle();
        flag = wxHSCROLL;
        text->SetWindowStyle(flag);

        //status:
        PushStatusText(wxT("Status: Setting text into one line..."));
        wxSleep(1.5);
        PopStatusText();

        wxTextCtrl* textNew = new wxTextCtrl(this, wxID_ANY, text-
>GetValue(), wxPoint(0, 0), text->GetSize(), flag | wxTE_MULTILINE);
        text->Destroy();
        text = textNew;
        textNew-
>SetFont(wxFont(11, wxFONTFAMILY_DEFAULT, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORM
AL, false, wxEmptyString));

        int w, h;
        GetVirtualSize(&w, &h);
        SetVirtualSize(w, h + 200);
        wxSize sz = GetSize();
        sz.SetHeight(sz.GetHeight() - 1);
        this->SetSize(sz);
        sz.SetHeight(sz.GetHeight() + 1);
        SetSize(sz);
    }

    void OnFormatWordWrap(wxCommandEvent& event) {
        wxTextCtrl* textNew = new wxTextCtrl(this, wxID_ANY, text-
>GetValue(), wxPoint(0, 0), text-
>GetSize(), wxTE_PROCESS_ENTER | wxTE_MULTILINE);
        text->Destroy();
        text = textNew;
        textNew-
>SetFont(wxFont(11, wxFONTFAMILY_DEFAULT, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORM
AL, false, wxEmptyString));

        wxSize sz = this->GetSize();
        sz.SetHeight(sz.GetHeight() - 1);
```

```
    this->SetSize(sz);
    sz.SetHeight(sz.GetHeight() + 1);
    this->SetSize(sz);

    //status:
    PushStatusText(wxT("Status: Word wrap"));
    wxSleep(1.5);
    PopStatusText();
}

void OnFormatFont(wxCommandEvent& event) {
    //status:
    PushStatusText(wxT("Status: Format font"));
    wxSleep(1.5);
    PopStatusText();

    wxFontDialog* fontDialog = new wxFontDialog(this);
    fontDialog->ShowModal();

    //status:
    PushStatusText(wxT("Status: Font is formated..."));
    wxSleep(1.5);
    PopStatusText();
}

void OnViewStatBar(wxCommandEvent& event) {
    if (this->GetStatusBar()->IsShown())
    {
        //status:
        PushStatusText(wxT("Status: Hide status bar"));
        wxSleep(1.5);
        PopStatusText();

        this->GetStatusBar()->Hide();
        isStatusShow = false;
    }
    else
    {
        this->GetStatusBar()->Show();
        isStatusShow = true;

        //status:
        PushStatusText(wxT("Status: Show status bar"));
        wxSleep(1.5);
        PopStatusText();
    }
    wxSize sz = this->GetSize();
    sz.SetHeight(sz.GetHeight() - 1);
    this->SetSize(sz);
    sz.SetHeight(sz.GetHeight() + 1);
    this->SetSize(sz);
}

void OnHelpAbout(wxCommandEvent& event) {
    wxAboutDialogInfo info;
    info.SetName("Oiri Notepad");
    info.SetVersion("1.1");
```

```
        info.SetDescription("Example of implementing the Notepad using wxWidget
s library");
        info.SetCopyright("(C) ajanach (2022.)");
        info.AddDeveloper("Antonio Janach");
        wxAboutBox(info, this);
    }

    void OnUpdateUIUndo(wxUpdateUIEvent& event) {
        event.Enable(text->CanUndo());
    }

    void OnUpdateUICut(wxUpdateUIEvent& event) {
        event.Enable(text->CanCut());
    }

    void OnUpdateUICopy(wxUpdateUIEvent& event) {
        event.Enable(text->CanCopy());
    }

    void OnUpdateUIPaste(wxUpdateUIEvent& event) {
        event.Enable(text->CanPaste());
    }

    enum MenuControls {
        idFileNew = 1000,
        idFileNewWindow = 1001,
        idFileSave = 1002,
        idFileSaveAs = 1003,
        idFilePageSetup = 1004,
        idFilePrint = 1005,
        idEditUndo = 1006,
        idEditCut = 1007,
        idEditCopy = 1008,
        idEditPaste = 1009,
        idEditDelete = 1010,
        idEditFind = 1011,
        idEditReplace = 1012,
        idEditGoto = 1013,
        idEditSelectAll = 1014,
        idEditTimeDate = 1015,
        idFormatAutoLine = 1016,
        idFormatWordWrap = 1017,
        idFormatFont = 1018,
        idViewStatBar = 1019,
        idHelpAbout = 1020,
        idFileOpen = 1021, idFileExit
    };

    DECLARE_EVENT_TABLE()
};

class MainApp : public wxApp {
public:
    virtual bool OnInit() {
        Notepad* main = new Notepad();
        main->Show(true);
        return true;
    }
```

```
};

BEGIN_EVENT_TABLE(Notepad, wxFrame) //početak event tablice za Notepad klasu:
//Kartica file:
EVT_MENU(idFileNew, Notepad::OnFileNew)
EVT_MENU(idFileNewWindow, Notepad::OnFileNewWindow)
EVT_MENU(idFileOpen, Notepad::OnFileOpen)
EVT_MENU(idFileSave, Notepad::OnFileSave)
EVT_MENU(idFileSaveAs, Notepad::OnFileSaveAs)
EVT_MENU(idFilePageSetup, Notepad::OnFilePageSetup)
EVT_MENU(idFilePrint, Notepad::OnFilePrint)
EVT_MENU(idFileExit, Notepad::OnFileExit)

//kartica edit:
EVT_MENU(idEditUndo, Notepad::OnEditUndo)
EVT_MENU(idEditCut, Notepad::OnEditCut)
EVT_MENU(idEditCopy, Notepad::OnEditCopy)
EVT_MENU(idEditPaste, Notepad::OnEditPaste)
EVT_MENU(idEditDelete, Notepad::OnEditDelete)
EVT_MENU(idEditFind, Notepad::OnEditFind)
EVT_MENU(idEditReplace, Notepad::OnEditReplace)
EVT_MENU(idEditGoto, Notepad::OnEditGoto)
EVT_MENU(idEditSelectAll, Notepad::OnEditSelectAll)
EVT_MENU(idEditTimeDate, Notepad::OnEditTimeDate)

//kartica format:
EVT_MENU(idFormatAutoLine, Notepad::OnFormatAutoLine)
EVT_MENU(idFormatWordWrap, Notepad::OnFormatWordWrap)
EVT_MENU(idFormatFont, Notepad::OnFormatFont)

//kartica view:
EVT_MENU(idViewStatBar, Notepad::OnViewStatBar)

//kartica about:
EVT_MENU(idHelpAbout, Notepad::OnHelpAbout)

//event na klik X gumba okvira:
EVT_CLOSE(Notepad::OnXButton)

//wxUpdateUIEvent:
EVT_UPDATE_UI(idEditUndo, Notepad::OnUpdateUIUndo)
EVT_UPDATE_UI(idEditCut, Notepad::OnUpdateUICut)
EVT_UPDATE_UI(idEditCopy, Notepad::OnUpdateUICopy)
EVT_UPDATE_UI(idEditPaste, Notepad::OnUpdateUIPaste)

END_EVENT_TABLE() // kraj event tablice

//implementacija glavnog programa:
IMPLEMENT_APP(MainApp)
```