Tehnički Fakultet “Mihajlo Pupin”  
Zrenjanin

Seminarski rad

Tema: Programiranje niskog nivoa u računarskim igricama

Predmet: Operativni sistemi  
Profesor: x.x. Student: x.x.

**Sadržaj**

Uvod ....................................................................................................................................................... [[3]](#Bookmark1)Istorijat i teorijske osnove ...................................................................................................................... [[4]](#Bookmark2)Analiza tehnoloških osnova ................................................................................................................... [[5]](#Bookmark3)Komparativna analiza postojećih rešenja ............................................................................................... [[8]](#Bookmark4)Realizovan primer ................................................................................................................................ [[10]](#Bookmark5) Namena softvera i korisnicko uputstvo ................................................................................. [[10]](#Bookmark5) Kratak opis primenjenog razvojnog alata(razvojnog okruženja) Code::Blocks ............................ [[10]](#Bookmark5) Lista naredbi korišćenih u realizovanom primeru igre “Iks-Oks”, u razvojnom okruženju  
 Code::Blocks ....................................................................................................................................[[11]](#Bookmark6) Opis implementacije …................................................................................................................... [[12]](#Bookmark7)Zaključak .............................................................................................................................................. [[25]](#Bookmark8)Literatura................................................................................................................................................. [[26]](#Bookmark9)

**Uvod**

-Cilj ovog rada je upoznavanje sa osnovnim terminima iz naslova, upoznavanje sa istorijom (sažetom)  
 programskih jezika,kao i istorijom (sažetom) računarskih igara, komparacija različitih razvojnih okruže-  
 nja za različite programske jezike (u kontekstu programiranja računarskih igrica), i realizovanje primera  
 programiranja računarske igre na niskom nivou.

-Istorijat i teorijske osnove- upoznavanje sa (sažetim) istorijatima programskih jezika i računarskih igara,  
 pojmovi računarske igrice, programskog jezika i razvojnog okruženja.  
  
-Analiza tehnoloških osnova- analiza programskih jezika na niskom nivou (u kontekstu programiranja  
 računarskih igrica), kratka objašnjenja nižih programskih jezika, mašinski kod, zašto su programi pi-  
 sani višim programskim jezicima, sporiji od programa pisanim nižim programskim jezicima.  
  
-Komparativna analiza postojećih rešenja- Kratak listing, opis i komparacija nekoliko razvojnih okruže-  
 nja i njihovih karakteristika, i komparacija mašinskog i asemblerskog jezika.  
  
-Realizovan primer- Namena softvera i korisničko uputstvo, kratak opis primenjeog razvojnog alata,  
 listing naredbi koje su korištene pri izradi softvera i opis implementacije i prikaz izvršavanja softvera.

-Zaključak- Šta je zaključeno tokom izrade rada.

-Literatura- Literatura koja je korištena tokom izrade rada.

**Istorijat i teorijske osnove**

Pojam računarske igrice:  
  
-Računarska igrica je interaktivno okruženje gde igrači moraju savladati izazove, tako što predu-  
 zimaju akcije određene pravilima, kako bi ispunili uslov ili uslove za pobedu.

Kratak istorijat računarskih igrica:  
  
-Prva računarska igra ikada napravljena je “Spacewar” 1962. , napravio ju je Stiven Rasel (Steven  
 Russel) koristeći vektorsku grafiku na PDP-1. Napisana je PDP-1 assembly jezikom. “Spacewar” je  
 takođe instaliran na više, primarno akadameksih računara, što znači da je prva igra koja je igrana na  
 više računara. [[1]](https://www.geeksforgeeks.org/history-of-computer-games/)-1972. “The Odyssey” postala je prva komercijalna kućna konzola za video igre, koju je u proizvodnju  
 pustio Magnavox. Sastojala se od bele, crne i braon kutije koja se povezivala sa televizorom, i dva  
 pravougaona kontrolera povezana žicama.[[1]](https://www.geeksforgeeks.org/history-of-computer-games/https://www.geeksforgeeks.org/history-of-computer-games/)-Ljudi su polako prepoznavali potencijal zabave i profita koji igrice mogu da donesu, time nastaje sve  
 više novih konzola za igre, novi koncepti, nove priče i tehnologija, i tokom godina se tehnologija i moć  
 računara izuzetno povećala i poboljšala.

Pojam IDE (Integrated Development Environment – Ingerisano razvojno okruženje)  
-Integrisano razvojno okruženje (IDE) je aplikacija koja olakšava razvoj druih aplikacija (u ovom slučaju  
 računarskih igara). [[5]](https://www.veracode.com/security/integrated-development-environment)

Pojam programskog jezika:  
-Programski jezik je način komuniciranja između programera i računara.  
-Programski jezik je skup komandi i instrukcija koje dajemo mašini, kako bi ona izvršila  
 određen zadatak. [[2]](https://www.geeksforgeeks.org/the-evolution-of-programming-languages/)-Programske jezike delimo na više i niže programske jezike. Razlika među njima je: [[3]](https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/)

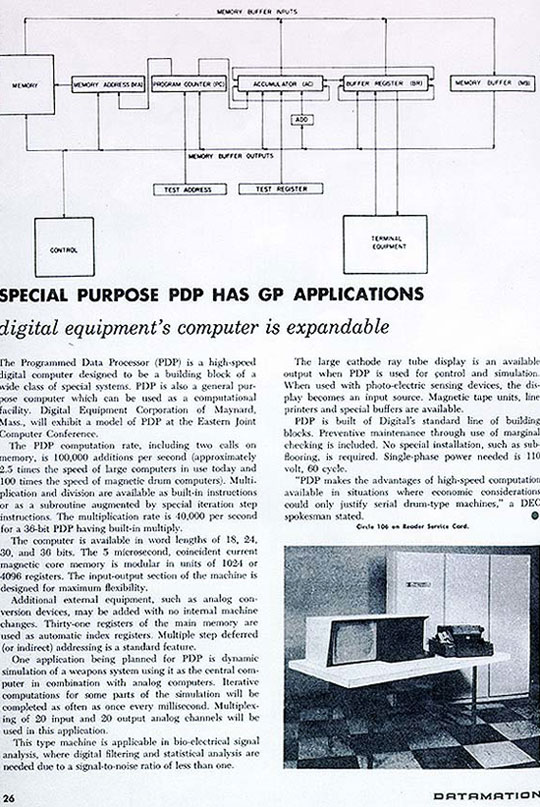
|  |  |
| --- | --- |
| Viši programski jezik | Niži programski jezik |
| “Prijateljski je nastrojen” programerima | “Prijateljski je nastrojen” mašini |
| Manja efikasnost memorije | Visoka efikasnost memorije |
| Lak je za razumevanje čoveku | Težak je za razumevanje |
| Proces debagovanja(debugging) je lak | Debagovanje(debugging) je kompleksno |
| Kod je lak za održavanje | Kod je kompleksan i težak za održavanje |
| Kod je portabilan | Kod nije portabilan već zavisi od mašine |
| Radi na bilo kojoj platformi | Mašinski zavistan |
| Kodu je potreban kompajler ili interpreter za prevođenje | Kodu je potrebna asembler za prevođenje |
| U širokoj je primeni za programiranje | Korišćenje nije često u današnjici |

**Analiza tehnoloških osnova**

Kratak istorijat i analiza programskih jezika:  
  
1949. Godina: ASEMBLER (ASSEMBLY)

-Pojava asemblerskog (assembly) jezika. To je tip programskom jezika niskog nivoa. Uglavnom se sasto-  
 ji od instrukcija(simbola) koje samo mašina može da razume. [[2]](https://www.geeksforgeeks.org/the-evolution-of-programming-languages/)

-U današnjici asemblerski (assembly) jezik se koristi za programe u realnom vremenu,  
 kao što su simulacija navigacije sistema pri letenju, medicinskoj opremi i drugo. Takođe se može  
 koristiti i za kreiranje računarskih virusa i drajvera.  
-Svaka jedna instrukcija u asemblerskom jeziku odgovara tačno jednoj instrukciji na mašinskom jeziku.  
-Prva računarska igra ikada napravljena, napravljena je pomoću asemblerskog (assembly) jezika  
 na PDP-1 računaru. [[1]](https://www.geeksforgeeks.org/history-of-computer-games/)

  
Slika preuzeta sa <https://history-computer.com/ModernComputer/Electronic/Images/pdp-1_datamation.jpg><https://history-computer.com/pdp-1/> [[4]](https://history-computer.com/pdp-1/)Na slici je dat prikaz i opis PDP-1 računara iz magazina DATAMATION 1959. Godine. [Slika 1.]

1957. godina: FORTRAN  
  
-Pojavljuje se FORTRAN. Napravili su ga Džon Bakus i IBM.  
-Dizajniran za rad sa numeričkim vrednostima i naučnim računarstvom(science computing).  
-U njemu je napisan softver za svemirske sonde Voyager-1 i Voyager-2 od NASA (softver originalno  
 pisan u FORTRAN 5 programskom jeziku). [[2]](https://www.geeksforgeeks.org/the-evolution-of-programming-languages/)-Većinska primena FORTRAN programskog jezika je u naučne svrhe, ali je takođe upotrebljen radi krei-  
 ranja video igara. Primer bi bila igra ”The Oregon Trail” iz 1971. godine. Napravili su je Don Rawitsch,  
 Bill Heinemann i Paul Dillenberger. Igra je napravljena za minikompjuter (HP 2100).  
  
1972. Godina: C

-Pojavljuje se C. To je proceduralni programski jezik opšte namene i jedan od najpopularnijih  
 programskih jezika. [[2]](https://www.geeksforgeeks.org/the-evolution-of-programming-languages/)-Veći deo koda koji je prethodno napisan u asemblerskom jeziku, zamenjen je C-om, kao  
 na primer operativni sistemi, jezgro (kernel) operativnog sistema, i mnoge druge aplikakacije. [[2]](https://www.geeksforgeeks.org/the-evolution-of-programming-languages/)-C ima široku primenu, na primer koristi se za razvoj sistemskih softvera, aplikativnih softvera i  
 ugrađenih (embedded) sistema.  
-Programski jezik C je veoma koristan za kreiranje Game Engine-a, ali nije baš popularan za kreiranje  
 samih igara, kakogod postoje primeri igara programiranih u C programskom jeziku. Neke od tih igara su  
 “Doom” 1993. godine, i kasnije u radu odrađen primer igre “Iks-Oks” specifično za ovaj rad [[Realizovan Primer].](#Bookmark1)

1991. Godina: VISUAL BASIC (VB)

-BASIC (Beginners All-purpose Symbolic Instruction Code) 1964. Godine.  
-Prva mikroračunarska verzija BASIC-a napisana je od strane Bil Gejts-a, Pol Alen-a i Monte  
 Davidov-a za svoju novo-formiranu kompaniju Microsoft.  
-1991. Godine Microsoft je izbacio Visual Basic, što je unapređena verzija BASIC-a. [[5]](https://www.veracode.com/security/integrated-development-environment)-Microsoft-ov Visual Basic je jedan od prvih programskih jezika sa integrisanim razvojnim okruženjem  
 (IDE).  
-Microsoft-ov Visual Basic nam je pokazao da se programiranje može zamisliti i u grafičkim  
 terminima.  
  
Zašto se pojavio IDE (Integrisano razvojno okruženje)?

-IDE se pojavio zbog povećanja produktivnosti, skraćenja vremena potrebnog za postavu (setup), pove-  
 ćanje brzine razvojnih zadataka, i najbitnije radi standardizacije razvojnog procesa.

Zašto su stare igre, čak iako su postojali jezici viši od assembly programskog jezika (poput C,  
FORTRAN i COBOL), idalje programirane u assembly jeziku?

-Igre su programirane u assembly jeziku zbog tehnoloških ograničenja, tj. jačine računara za koje su  
 programirane.  
  
-Tada su se igre nalazile na karticama sa veoma malom količinom memorije, 8-bitni mikroproceso-  
 rski čipovi nisu mogli da podrže tu količinu memorije koja je bila potrebna višim jezicima.  
-Pored toga bitna je bila i performansa, tj. bilo je bitno iskoristiti, što je moguće bolje, hardver koji je na  
 raspolaganju. [[7]](https://stackoverflow.com/questions/4904707/why-were-old-games-programmed-in-assembly-when-higher-level-languages-existed)

\*Ovaj princip stoji čak i danas. Postoje jezici kao što su Python-a i C#, sa lakom i razumljivom sinta-  
 ksom, ali zbog njihove zahtevnosti, odnosno količine memorije koje ,programi koji su programirani  
 ovim jezicima, zauzmu, koristi se jezik kao što je C++. Programski jezik C++ nije toliko hardverski  
 zahtevan, ali sintaksa programskog jezika jeste malo komplikovavnija od sintakse C# i Python.  
 C++ pruža mnogo bolju performansu od ostalih jezika.\*

Zašto su programi napisani višim programskim jezicima sporiji od nižih programskih jezika?

-Programi pisani višim programskim jezicima su sporiji, tj. imaju manje performanse, jer je potrebno  
 više puta prevesti jednu komandu u mnoštvo drugih, za svaki nivo udaljenosti od mašinskog koda. [[8]](https://stackoverflow.com/questions/68596666/why-high-level-language-is-considered-slower-than-lower-level)-Sumirano: Što je programski jezik više apstraktan (dalji od mašinskog koda), on je sporiji jer jedna  
 linija koda u tom jeziku je ekvivalentna mnoštvu na mašinskom jeziku.

Šta je mašinski kod, odnosno mašinski jezik? [[9]](https://www.javatpoint.com/what-is-machine-language)

-Mašinski kod / objektni kod / mašinski jezik je najniži oblik “programskog jezika”. To je jezik samog  
 hardvera računara, i jedini jezik koji računar razume. Podaci su u mašinskom kodu predstavljeni  
 nulama i jedinicama (0 i 1).  
  
-Svaki oblik podatka, instrukcije, ili bilo čega drugog, u računaru se predstavlja pomoću nule i jedinice.

-Svaki program napisan bilo kojim programskim jezikom, a da nije mašinski, mora se prevesti putem  
 programskih prevodioca, u objektni kod, odnosno mašinski kod, koji računar može da razume.

**Komparativna analiza postojećih rešenja**

-Danas postoji mnogo razvojnih orkuženja, a najpoznatiji, u kontekstu mogućnosti kreiranja igara, od  
 njih su: [[6]](#Bookmark16)

* Visual Studio
* CLion
* Eclipse
* Code::Blocks
* CodeLite
* SASM

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Cena | Podržan na operativnim sistemima | Debager (Debugger) | Podržani programski jezici | Zahtevnost hardvera |
| Visual Studio | Besplatan | Windows, macOS i Linux | Integrisan | HTML,CSS,JavaScript, C#,Python, PHP, C, C++, Asembler | Zahtevan, može biti i veoma zahtevan |
| CLion | Besplatan, mogućnost kupovine zbog dodatnih opcija | Windows, macOS i Linux | Integrisan | Python, Objective-C/C++, HTML,CSS,JavaScript, XML | Nije mnogo zahtevan ali zahtevniji je od drugih |
| Eclipse | Besplatan | Windows, macOS i Linux | Integrisan | Java, (pomoću ekstenzija moguća je podrška raznih jezika) | Nije zahtevan |
| Code::Blocks | Besplatan | Windows, macOS i Linux | Integrisan (ako se skine verzija sa već ubačenim debagerom) | Primarno C,C++, ali mogući su i Java, Python, HTML5, Objective-C, PHP, Ada, Fortran, Assembler,.. | Nije zahtevan |
| CodeLite | Besplatan | Windows, OSX, Linux, Ubuntu /Debian, Fedora / OpenSUSE,macOS,FreeBSD | Integrisan | C, C++, PHP i Node.js | Nije zahtevan |
| SASM | Besplatan | BSD, Windows, Linux | Integrisan | Asembler(MASM,NASM,GAS, FASM) | Nije zahtevan |
| SimplyFortran | 30 dana besplatna proba, nakon toga se mora kupiti | Windows, macOS, Linux | Integrisan | Asembler, BASIC, C/C++, Fortran, Makefile, Lua, MsDOS/Windows Batch, Python,HTML, Tcl, UNIX-like Shells | Nije zahtevan |

Kratak opis navedenih razvojnih okruženja:

* Visual Studio - je IDE sa širokom podrškom programskih jezika, primarni C#, besplatan, ima integrisan debager ali je hardverski mnogo zahtevan, može zahtevati između 2-14 ili više GB RAM memorije. Ima veliku mogućnost menjanja izgleda i stilizovanja.
* CLion – je IDE sa širokom podrškom programskih jezika, primarno C/C++, besplatan (mogućnost kupovine zbog dodatnih opcija), ima integrisan debager, nije mnogo hardverski zahtevan ali je zahtevniji od ostalih navedenih razvojnih okruženja. Mogućnost menjanja izgleda (biranje između tri početne teme, moguće je, skidanjem plugin-a imati i više tema).
* Eclipse – je skalabilno, besplatno, poslovno okruženje otvorenog koda (Open Source) sa podrškom Java programskog jezika (moguće i drugih pomoću ekstenzija). Moguće je menjanje izgleda boje/teme. Hardverski nije zahtevan.
* Code::Blocks – besplatan IDE, sa primarnom podrškom programskog jezika C i C++ (Moguće i drugih), ima integrisan debager (i kompajler ako se skine verzija sa kompajlerom), menjanje izgleda je veoma ograničeno, hardverski nije zahtevan.
* CodeLite – besplatan IDE sa integrisanim debagerom, podržava programske jezike C, C++, PHP i Node.js, hardverski nije zahtevan. Menjanje izgleda je moguće (skidanjem XML fajla i ubacivanjem u okruženje)
* SASM – besplatan IDE sa integrisanim debagerom, podržava MASM, NASM, GAS i FASM asemblerske jezike. Hardverski nije zahtevan.
* SimplyFortran – besplatan (na 30 dana, posle toga se plaća) IDE sa integrisanim debagerom, dizajniran je za programski jezik Fortran (ali podržava i neke druge jezike), hardvevrski nije zahtevan.

Komparacija mašinskog koda i assembly programskom jezika [[10]](https://www.geeksforgeeks.org/difference-between-machine-language-and-assembly-language/)

|  |  |
| --- | --- |
| Mašinski jezik | Asemblerski (assembly) jezik |
| Razumljiv samo računaru | Razumljiv samo ljudima |
| Podaci su predstavljeni u binarnom formatu, odnosno sa 0 i 1. | Podaci su predstavljeni mnemonicima poput Mov, ADD, SUB,... |
| Modifikacija i popravljanje grešaka je nemoguće | Modifikacija i popravljanje grešaka je moguće |
| Izvršavanje programa je izuzetno brzo jer su svi podaci već prisutni u svojoj binarnoj formi | Izvršavanje je komparativno sporije |
| Nema potrebe za ikakvim prevodiocem, mašinski kod jeste jezik mašine | Potreban je programski prevodioc kako bi mašina razumela mnemonike |

**Realizovan primer**

Namena softvera i korisničko uputstvo

-Za ovaj rad je konkretno realizovan primer igre “Iks-Oks".  
-Namena softvera je simulacija klasične igre “Iks-Oks” pomoću programskog jezika C i razvojnog  
 okruženja (IDE) “Code::Blocks”.

-Kada korisnik prvi put pokrene program, otvara se komandna linija u kojoj se ispisuje glavni, odnosno  
 početni meni, i kratko uputstvo šta bi korisnik od tastera trebao da pritisne, kako bi izašao iz programa,  
 odnosno započeo novu, ili nastavio staru igru.

-Pritiskom na taster zadužen za novu igru ili nastavak stare, korisnik dobija nov meni, koji sadrži tablu  
 na kojoj će se dalje odvijati igra. U ovom meniju se takođe nalaze smernice (kratke poruke), o tome šta  
 bi korisnik od tastera trebao pritisnuti za dalje izvršenje (upis znaka X ili O na polje table) ili izlazak  
 iz programa. Pored table se takođe nalazi prikaz brojača dosadašnjih pobeda za oba znaka.

-U slučaju da je korisnik uspeo da pobedi (ispunio uslov da se tri polja, sa istim znakom, mogu povezati  
 pravom linijom), program kroz konzolu ispisuje nov meni, koji prikazuje aktuelno stanje table,i tri polja  
 (pobednička polja) povezana pravom linijom, kao i poruku u kojoj piše koji znak (X ili O) je pobedio.  
 Isto tako se ispisuje i uputstvo o zatvaranju, odnosno prestanku rada programa.

-U slučaju da je došlo do toga da je rezultat igre nerešen, program ispisuje meni zadužen za ispis  
 aktuelnog stanja table, kao i poruke da je rezultat nerešen, i ispis kratke smernice o prestanku rada  
 programa.  
-Svaki od menija je zadužen za ispis naslova igre “Iks-Oks”.

Kratak opis primenjenog razvojnog alata (razvojnog okruženja) Code::Blocks

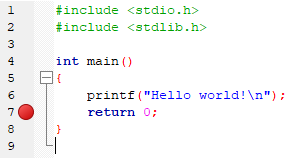
-Code::Blocks je besplatno razvojno okruženje, sa ugrađenim debagerom. Koristimo GNU GCC  
 Compiler.  
-Razvojno okruženje Code::Blocks podržava, primarno, programske jezike C i C++.  
-Da bi radili (pisali program) u Code::Blocks-u, prvo moramo kreirati projekat. U ovom primeru  
 kao kategoriju projekta koristimo konzolnu aplikaciju (Console Application), i izabran programski jezik  
 je C.

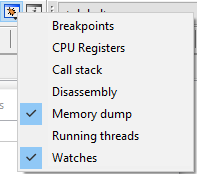
Prikaz opcija i dugmića za rad u razvojnom okruženju Code::Blocks [Slika 3.]

-Nakon što kreiramo projekat, možemo da počnemo sa razvojem našeg programa. Kada želimo  
 da pokrenemo naš program kliknemo na žuti zupčanik da bi build-ovali (build - razvojno okru-  
 ženje prevodi kod sa našeg nivoa, na niži nivo gde linker spaja potrebne biblioteke i loader  
 ubacuje naš program u memoriju) naš program. Ako nema grešaka (debugger - nam nije  
 prijavio nikakvu grešku) možemo pokrenuti izvršavanje našeg programa pomoću zelenog trou-  
 gla/strele pored zupčanika što će nam na ekranu prikazati konzolu u kojoj vidimo prikaz našeg  
 programa (možemo pritisnuti dugme build & run, što je dugme koje sadrži i zupčanik i trou-  
 gao).

\*To što je naš program izvršio build-ovanje bez prijave greške, ne mora da znači da naš program radi ispravno, i da radi ono što želimo (debugger proverava sintaksu i tipove promenljivih, ne konkretno ideju programera o datom programu)\*

-Crveni trougao na [Slici 3.] Je Debug/Continue dugme koje nam omogućava da izvršimo debaging  
 (debugging) našeg programa. Uz pomoć breakpoint-a možemo privremeno zaustaviti izvršavanje našeg  
 programa na onoj liniji koda na kojoj se nalazi breakpoint. Pomoću dugmića desno od trougla (Step into,  
 Step out, Next line,..) možemo se kroz naš kod (u debug modu) kretati, liniju po liniju, ulaziti i izlaziti  
 iz petlji.

 -Prikaz breakpoint-a (Crvena tačka na [Slici 4.]). Program se  
 na toj liniji koda zaustavlja i čeka dozvolu za dalje izvršavanje  
 (s tim što se ne izvrši ni ta linija koda na kojoj se nalazi  
 breakpoint, dok se ne nastavi dalje izvršavanje koda).  
  
 [Slika 4.]

-U razvojnom okruženju Code::Blocks je moguće prikazati  
 i izgled programa (vrednosti promenljivih) u memoriji, kao  
 i praćenje vrednosti samih promenljih pomoću prozora  
 ”Memory dump” i ”Watches”.  
 -Isto tako je moguć prikaz registara procesora (”CPU  
 Registers”), prikaz svih breakpoint-a (”Breakpoints”),  
 prikaz steka (”Call Stack”), prikaz niti (”Running  
 threads”), kao i proces disasembliranja (“Disassembly“).  
  
Slika [Slika 5.]

Lista naredbi korišćenih u realizovanom primeru igre “Iks-Oks” u razvojnom okruženju Code::Blocks

 -Biblioteku <stdio.h> koristimo za standardni ulaz-izlaz (I/O)  
 -Biblioteka <string.h> se koristi za rad sa string-ovima  
 -Biblioteku <stdbool.h> koristimo za promenljivu tipa bool (boolean)  
 [Slika 6.]

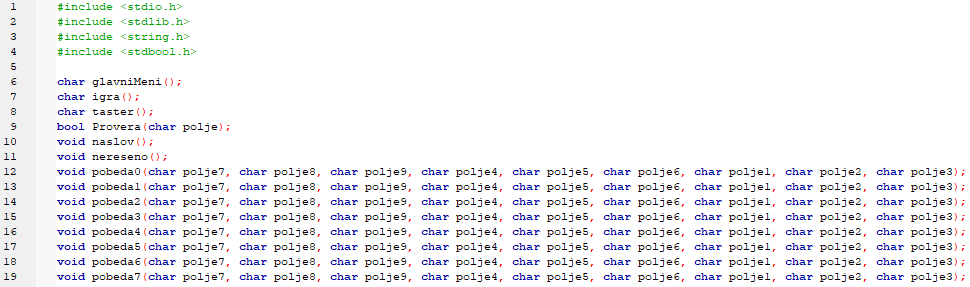
-Naredbu if() koristimo da bi proverili neki uslov, switch je kraća verzija takozvanog ugnježdenog  
 (nested) if-a.  
-Naredbu else koristimo u slučaju da uslov naredbe if nije ispunjen, i želimo da proverimo neki drugi  
 uslov.  
-Switch funkcioniše na osnovu date promenljive, prema stanju promenljive daje drugačije rezultate za  
 različite slučajeve (case).  
  
-Petlju (loop) for() koristimo kada znamo tačno određen broj ponavljanja neke akcije.  
-Petlju while() koristimo za ponavljanje iste akcije dok god se ne ispuni uslov za prekid petlje.  
-Naredba break se koristi kako bi nasilno izašli iz kontrole toka (for,if,switch,while).

-Naredbu getch() koristimo kako bi dobili unos sa tastature u ASCII obliku.  
-printf() koristimo za standardni ispis u konzoli.  
-system(“cls”) se koristi za brisanje podataka u konzoli (čišćenje konzole/osvežavanje).  
  
-Pomoću FILE \*fp kreiramo pokazivač (promenljivu koja pokazuje na memorijsku adresu druge pro-  
 menljive) na fajl (datoteku).  
-Naredbu fopen() koristimo za otvaranje određene datoteke u određenom režimu.  
-fclose() koristimo za zatvaranje datoteke kako bi oslobodili mesto u memoriji i omogućili rad datoteke  
 u nekom drugom režimu.  
-fprintf() naredbu koristimo za upis podataka u datoteku.  
-fscanf() naredbu koristimo za čitanje podataka iz datoteke.  
  
-return koristimo za vraćanje promenljive koja je istog tipa kao i funkcija iz koje se vraća data vrednost.

Tipovi promenljivih koji su korišćeni radi realizacije primera igre:

- char – Promenljiva tipa karakter, čuva jedan bajt u memoriji  
- int – Celobrojni tip promenljive, čuva vrednost celog broja (i pozitivan i negativan)  
- bool – Boolean tip promenljive, čuva vrednost true ili false, 0 ili 1.  
  
-U realizaciji primera takođe su korišćene funkcije i nizovi.  
-Funkcije tipa void ne vraćaju nikakvu vrednost, u ovom primeru korištene su za ispis rešenja.

Opis implementacije

Prikaz sekcije koda gde se nalaze biblioteke(Header files) i deklaracije funkcija [Slika 7.]

-U prvoj sekciji koda nalaze se biblioteke(Header files) i deklaracije funkcija.  
-Biblioteke(Header files) su tu da nam omoguće standardni ulaz-izlaz(I/O), rad sa stringovima i prome-  
 nljivama tipa boolean(bool).

-Funkcija main() poziva druge funkcije i beleži(ili briše) podatke o pobednicima u datoteci “rezultat.txt”.  
-Funkciju glavniMeni() koristimo za ispis glavnog menija, gde korisniku nudimo nekoliko opcija.  
 Pored toga što pruža opcije, ispisuje naziv igre (poziva funkciju naslov() zadužene za ispis naslova) i  
 ispisuje uputstvo za odabir opcije.

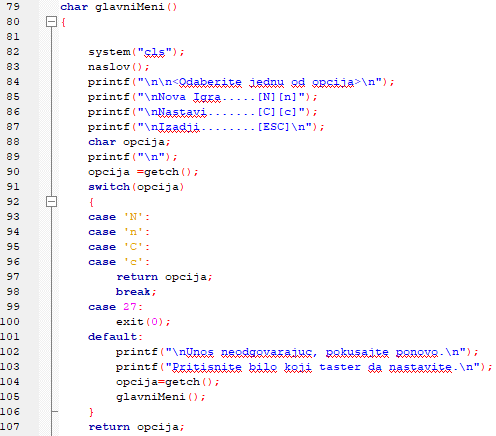
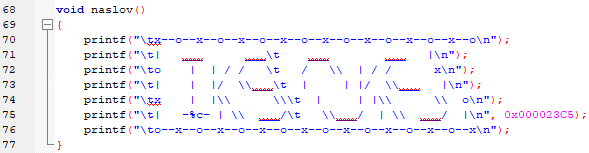
-Opcije:

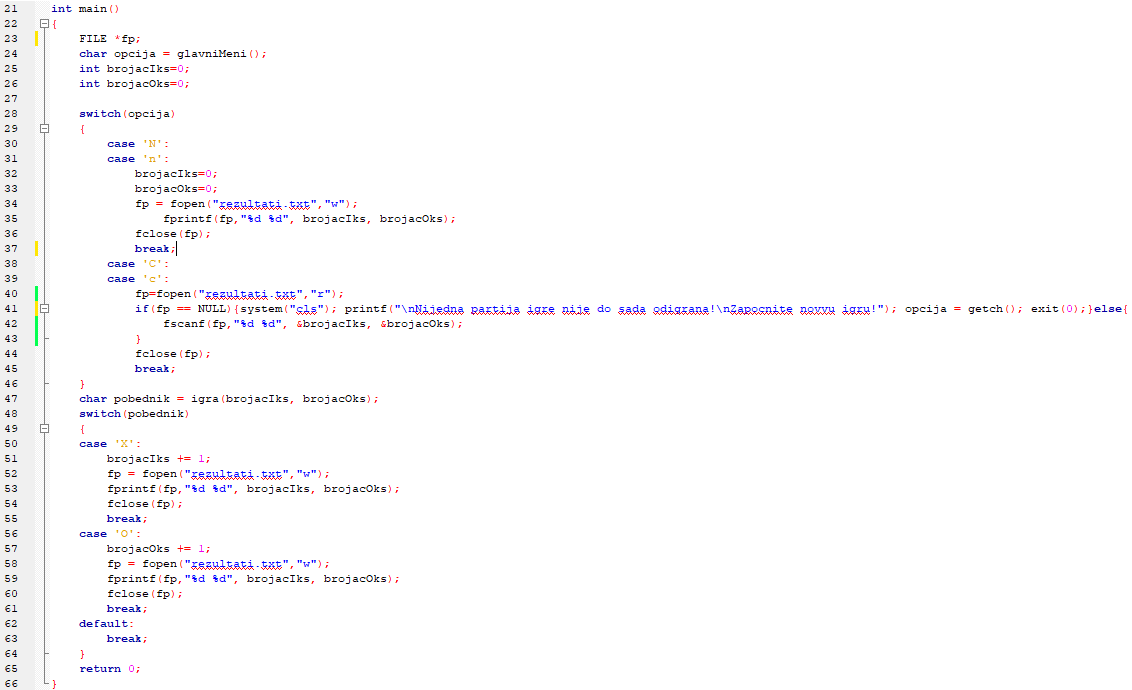
* Nova igra [N][n]
* Nastavi [C][c]
* Izađi [ESC]

-Pritiskom na taster “N” glavniMeni() vraća odabran slučaj, tj. vrednost tastera koji je korisnik na ta-  
 staturi pritisnuo, funkciji main(), isto tako se postupa i sa ostalim ponuđenim tasterima, osim tastera  
 “ESC”. U slučaju da je pritisnut taster “ESC” rad programa se prekida pomoću naredbe exit(0).

\*U slučaju da je na tastaturi pritisnut taster koji nije u ponudi, program korisniku vraća poruku o grešci i traži ponovni unos, i tako dok god unos nije zadovoljavajuć. To je moguće jer je funkcija glavniMeni() rekurzivna funkcija!!\*

-Dalje main() funckija preko naredbe switch() odlučuje šta će se desiti. Ako je pritisnut taster “N” ili  
 taster “C”.

  
Prikaz koda funkcije glavniMeni(), zadužene za poziv funkcije naslov() i ispis glavnog menija, kao i ispisom uputstva za odabir opcija koje program nudi [Slika 8.]  
  
  
Prikaz koda funkcije naslov(), zadužene za ispis naslova igre [Slika 9.]

Prikaz funkcija main() zadužene za poziv funkcije glavniMeni() i poziv funkcije igra() [Slika 10.]

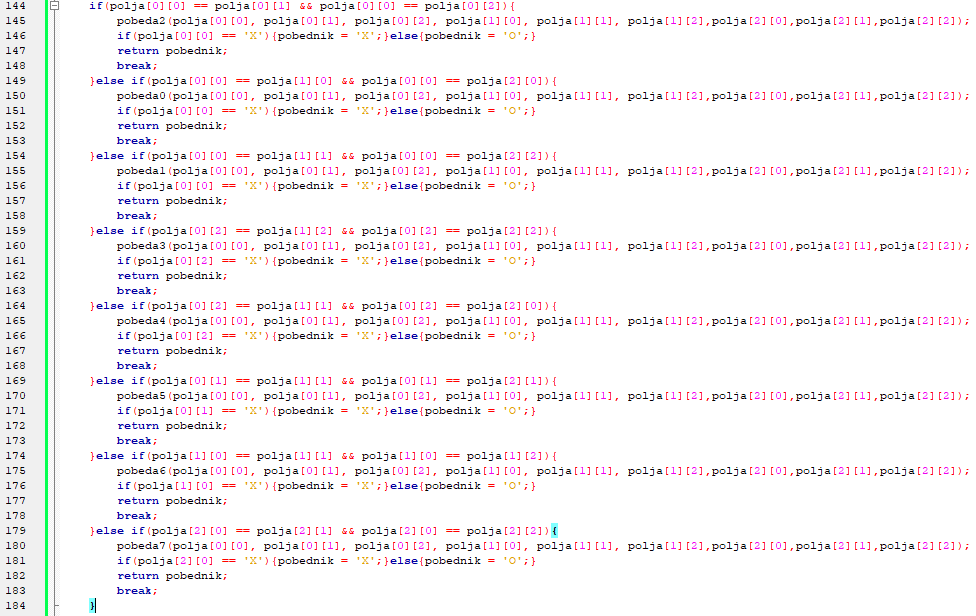
-Pored toga što je zadužena za poziv funkcije glavniMeni(), funkcija main() je zadužena za poziv funkci-  
 je u kojoj se odvija sama igra, to je funkcija igra().  
  
-Funkcija main() funkciji igra(), daje prave parametre brojacIks i brojacOks (brojači zaduženi za praćenje  
 broja pobeda igrača X i igrača O). Te parametre funkcija igra() koristi za ispis aktuelnog broja pobeda  
 oba igrača.  
  
-Kada se završi radnja funkcije igra(), funkcija main() dobija vrednost o igraču koji je pobedio, i zatim  
 na osnovu pobednika, inkrementuje brojač pobednika, i nove aktuelne rezultate upisuje u datoteku  
 “rezultati.txt”.

\*U slučaju da igra ni jednom do sada nije odigrana, korisnik dobija poruku da mora započeti novu  
 igru(jer program nema datoteku iz koje može čitati rezultate prethodnih igara)\*

-Funkcija igra() je funkcija zadužena za izvršavanje toka same igre, praćenje broja rundi, praćenje uslova  
 pobede, takođe je zadužena za pozivanje drugih funkcija ( taster(), Provera(), nereseno (),  
 pobeda0(), pobeda1(), pobeda2(), pobeda3(), pobeda4(), pobeda5(), pobeda6(), pobeda7() ).

Prikaz prve sekcije koda funkcije igra() [Slika 11.]

-U prvoj sekciji koda funkcije igra() (Slika 7.), vidimo deklaraciju brojača runda=0, brojač rundi koristi-  
 mo u slučaju da je broj rundi prešao 8, u tom slučaju je nemoguće da ijedan igrač pobedi, što znači da  
 je u pitanju nerešen slučaj. Tada funkcija igra() čisti(briše podatke) konzolu, poziva funkciju naslov(),  
 pored toga ispisuje trenutno stanje table aktuelne igre, zatim poziva funkciju nereseno() i funkciji main()  
 vraća vrednost NULL, što znači da se neće ni jedan od brojača (brojacIks i brojacOks) inkrementovati, i  
 time će rezultati ostati isti.  
  
-Niz “polja[3][3]” sadrži podatke o poljima table igre. U njega upisujemo X ili O u zavisnosti od broja  
 runde (još jedna primena brojača rundi). Mesto slova X ili O zavisi od toga da li je to mesto na tabli već  
 zauzeto (da li je neko već upisao slovo na to mesto) i unosa broja polja na tastaturi.  
-Takođe u prvoj sekciji vidimo i izgled same table, i pored table vidimo brojač pobeda za oba igrača.  
-Pored toga vidimo i promenljivu dugme, koja je zadužena za čuvanje povratne vrednosti funkcije za  
 unos sa tastature, a to je funkcija taster().  
-Isto tako vidimo i promenljivu t koja je zadužena za čuvanje povratne vrednosti funkcije Provera(),  
 a funkcija Provera() je zadužena za proveru toga da li je polje koje je igrač odabrao, već zauzeto.

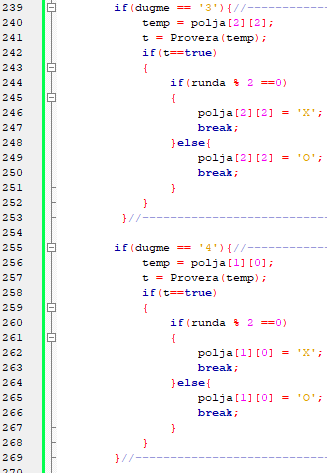
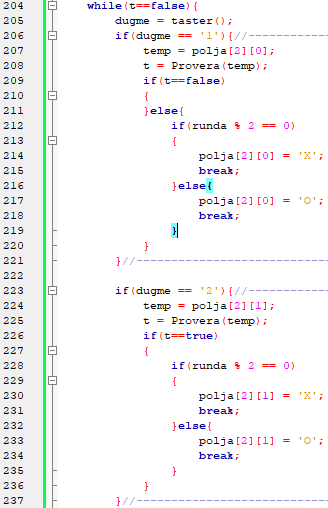
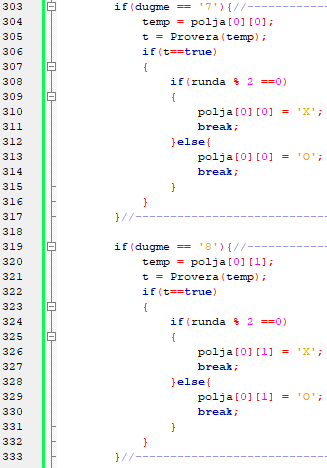
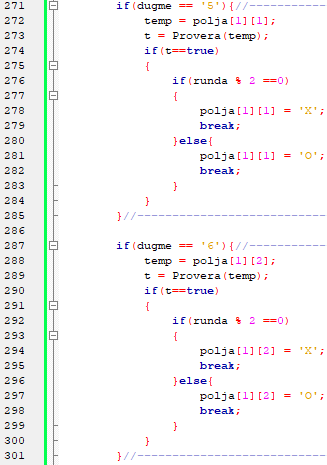
Prikaz druge sekcije koda funkcije igra() [Slika 11.]

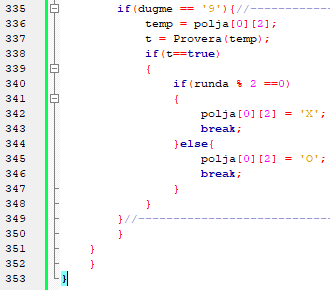
-U drugoj sekciji koda funkcije igra() (Slika 8.) vidimo proveru, koja se izvršava svake runde (za svaki  
 ciklus petlje for() ), da li je igrač X ili igrač O pobedio na jedan od 8 mogućih načina pobede.  
  
-Ako se bilo koja tri polja na tabli, s tim da se na njima nalazi isti znak/slovo, mogu spojiti pravom  
 linijom, to znači da je igrač sa tim slovom/znakom pobedio. U tom slučaju se poziva odgovarajuća  
 funkcija odgovorna za ispis aktuelnog stanja table, ali sa crtom koja povezuje tri odgovarajuća (pobe-  
 dnička) polja. Nakon toga se funkciji main() vraća vrednost (tj. karakter) pobednika i pomoću naredbe  
 break; se izlazi iz for() petlje i time završava rad funkcije igra() i samim tim i rad programa.  
  
-Mogući načini pobeda su:

* Tri uzastopna vertikalna polja sa istim znakom/slovom
* Tri uzastopna horizontalna polja sa istim znakom/slovom
* Tri uzastopna kosa (dijagonalna) polja sa istim znakom/slovom

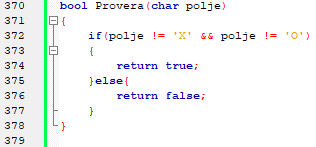
-Funkcije za ispis aktuelnog stanja table u odnosu na tip pobede:

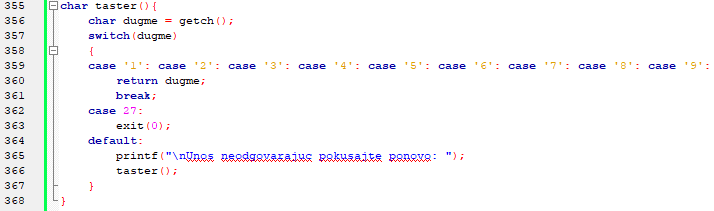
* pobeda0(), pobeda1(), pobeda2(), pobeda3(), pobeda4(), pobeda5(), pobeda6(), pobeda7()

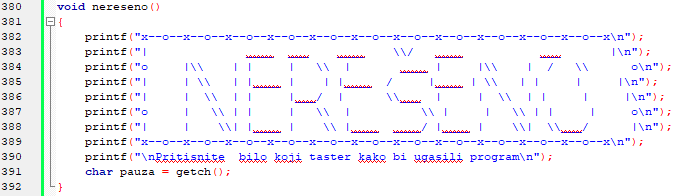
  


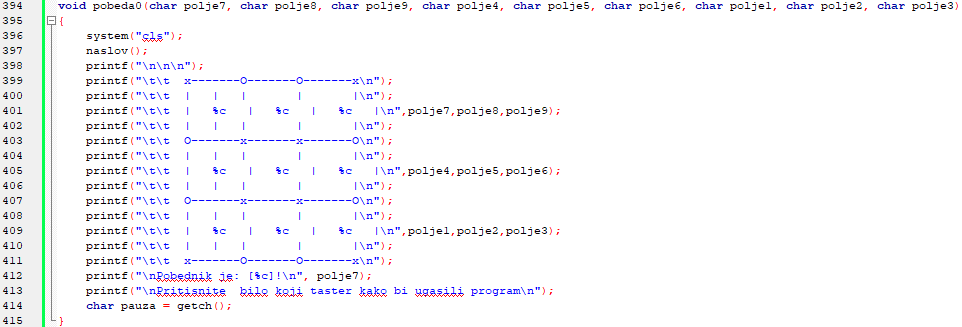
 -Na slikama [12.,13.,14.,15.,16.] prikazana je  
 treća sekcija koda funkcije igra(), zadužena  
 za unos i proveru, sa tastature, polja  
 na tabli.  
  
 -[Slika 9.] prikazuje početak while() petlje,  
 gde se proverava da li je promenljiva t  
 tačna ili netačna (true ili false). Dok god  
 je promenljiva t u stanju false while petlja  
 se izvršava. Vrednost promenljive t je  
 određena povratnom vrednošću funkcije  
 Provera(). Kada je promenljiva t u stanju  
 true, u zavisnosti od broja runde, polje  
 čiji je broj unet na tasteru tastature,  
 menja svoje stanje iz broja polja u znak  
 X, odnosno O. Zatim se iz while() petlje  
 izlazi, i kreće sledeci ciklus for() petlje.

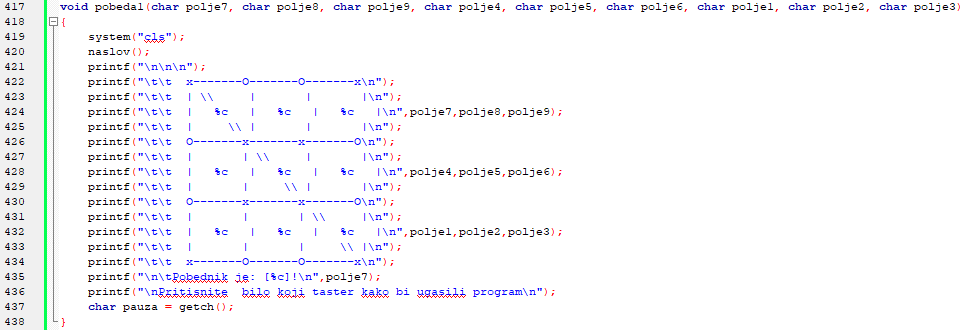
Prikaz treće sekcije koda funkcije igra() [Slike 12.,13.,14.,15.,16.]

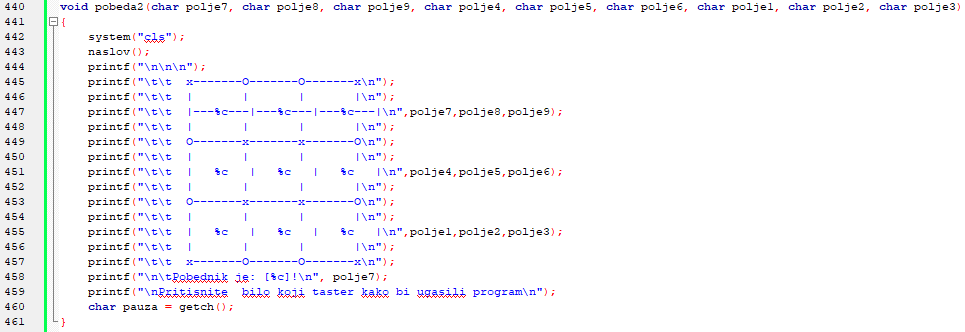
-Funkcija Provera() zadužena je za proveru toga da li je neko polje zauzeto ili ne. U slučaju da je zauzeto  
 povratna vrednost funkcije biće false, što znači da while ciklus funkcije igra() ponovo ide na nov ciklus,  
 a slučaju da to neko polje nije zauzeto, povratna vrednost funkcije Provera() biće true, što znači da to  
 određeno polje u funkciji igra() dobija vrednost X ili O.  
  
Prikaz funkcije Provera() [Slika 17.]

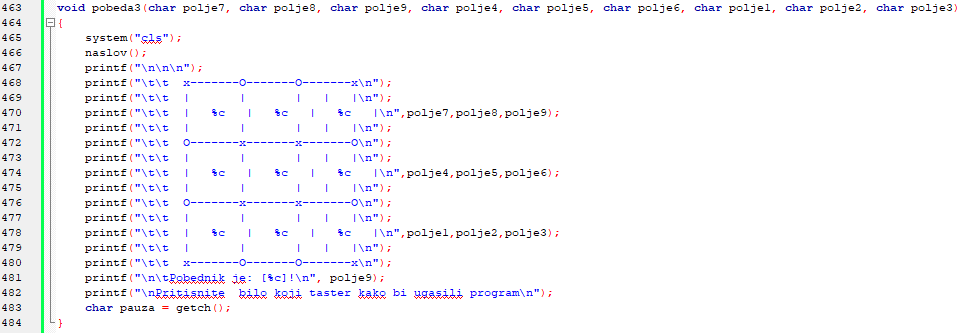
  
Prikaz koda funkcije taster() zadužene za vraćanje unosa, sa tastature, funkciji igra()[Slika 18.]

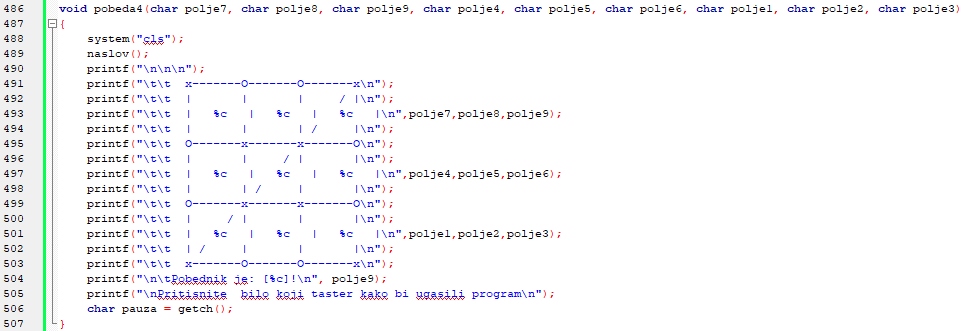
  
Prikaz koda funkcije nereseno() zadužene za ispis teksta NEREŠENO na ekranu. [Slika 19.]

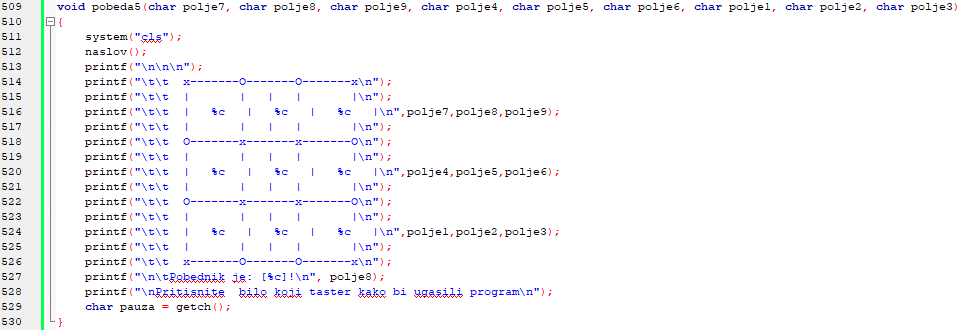
Prikaz koda funkcije pobeda0() zadužene za ispis table sa crtama koje povezuju “pobednička“ polja. [Slika 20.]

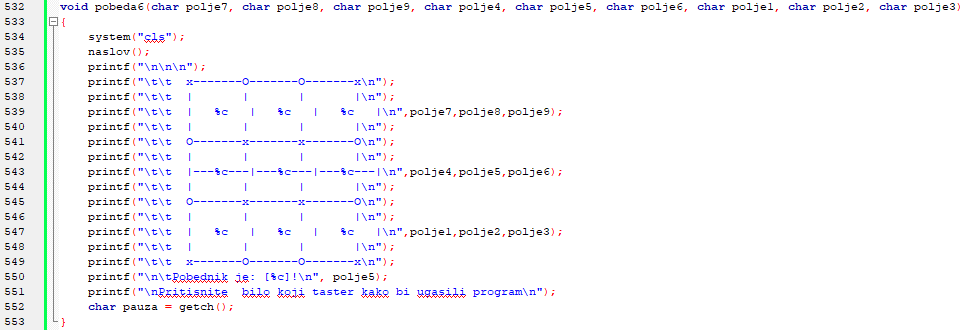
Prikaz koda funkcije pobeda1() [Slika 21.] zadužene za ispis table sa crtama koje povezuju “pobednička“ polja.

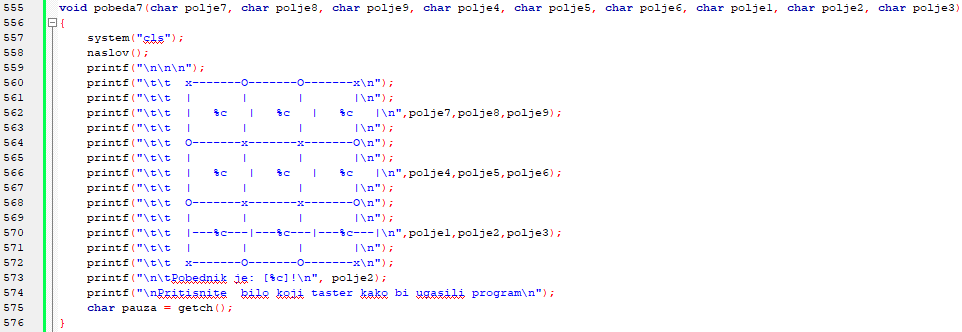
Prikaz koda funkcije pobeda2() zadužene za ispis table sa crtama koje povezuju “pobednička“ polja. [Slika 22.]

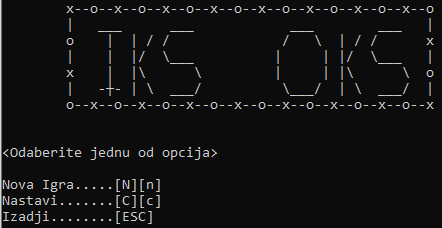
Prikaz koda funkcije pobeda3() zadužene za ispis table sa crtama koje povezuju “pobednička“ polja. [Slika 23.]

Prikaz koda funkcije pobeda4() zadužene za ispis table sa crtama koje povezuju “pobednička“ polja. [Slika 24.]

Prikaz koda funkcije pobeda5() zadužene za ispis table sa crtama koje povezuju “pobednička“ polja. [Slika 25.]

Prikaz koda funkcije pobeda6() zadužene za ispis table sa crtama koje povezuju “pobednička“ polja. [Slika 26.]

Prikaz koda funkcije pobeda7() zadužene za ispis table sa crtama koje povezuju “pobednička“ polja. [Slika 27.]

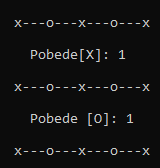
  
Prikaz pokretanja programa (realizovan primer) [Slika 28.]

  
Prikaz odabira opcije ”Nova Igra”, vidimo desno od table da su brojaci pobeda na nuli [Slika 29.]

  
Pobednik je X [Slika 30.]

  
Ponovno izvršavanje ali ovaj put je odabrana opcija ”Nastavi”, i vidimo da je brojač pobeda igrača X inkrementovan [Slika 31.]

  
Pobednik je O [Slika 32.]

  
Ponovno izvršavanje, opcija ”Nastavi”, vidimo da je svaki igrač pobedio jednom [Slika 33.]

**Zaključak**

**-**Ovim radom prikazana je mogućnost programiranja računarskih igrica na niskom nivou.  
-Programiranje računarskih igrica na jezicima niskog nivoa jeste moguće, što ne znači da bi trebalo da  
 zameni programiranje računarskih igrica na jezicima višeg nivoa.  
 Jezici nižeg nivoa jesu memorijski više efikasni, ali vremenski je zahtevnije (više vremena je potrebno)  
 programirati igricu na niskom nivou, nego programirati igricu na višem.  
 Zbog višeg nivoa apstrakcije viših programskih jezika, programi pisani njima su sporiji od programa  
 koji su pisani nižim programskim jezikom (odnosno zbog sintakse višeg programskog jezika, potrebno  
 je više puta prevoditi sa nivoa na nivo, sve do mašinskog koda, jednu instrukciju, i ta jedna instrukcija je  
 jednaka mnoštvu instrukcija na mašinskom kodu), ali, vreme potrebno za samo pisanje programa je  
 kraće, a u današnjici se ceni vreme programera više nego performansa programa, što znači da su  
 viši programski jezici efikasniji po tom pitanju, jer kod nižih jezika je komplikovanija sintaksa, tj.  
 sintaksa je bliža mašinskom kodu, što znači da nam je potrebno više linija koda za isti program pisan  
 višim programskim jezikom.  
 Takođe je kod, pisan nižim programskim jezikom, teži za održavanje, i otežano je otklanjanje grešaka.

**Literatura**

[1] Sa interneta – istorija računarskih igara  
<https://www.geeksforgeeks.org/history-of-computer-games/>

[2] Sa interneta – evolucija programskih jezika  
https://www.geeksforgeeks.org/the-evolution-of-programming-languages/

[3] Sa interneta – razlika između viših i nižih programskih jezika  
<https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/>

[4] Sa interneta (Slika strane magazina DATAMATION) - Slika strane magazina DATAMATION, opis  
računara PDP-1.  
https://history-computer.com/ModernComputer/Electronic/Images/pdp-1\_datamation.jpg  
Stranica na kojoj je slika pronađena:  
<https://history-computer.com/pdp-1/>

[5] Sa interneta – Sa stranice Veracode, definicija razvojnog okruženja i Visual Basic  
<https://www.veracode.com/security/integrated-development-environment>

[6] Sa interneta – Sa stranica su preuzeti opisi i karakteristike nekih programskih jezika  
<https://www.geeksforgeeks.org/10-best-ides-for-c-or-cpp-developers-in-2021/><https://visualstudio.microsoft.com/>

[7] Sa interneta – Preuzeto kratko “objasnjenje/citiranje”  
<https://stackoverflow.com/questions/4904707/why-were-old-games-programmed-in-assembly-when-higher-level-languages-existed>

[8] Sa interneta – Preuzeta ali i veoma izmenjena rečenica o tome zašto su viši programski jezici  
sporiji od nižih programskih jezika  
<https://stackoverflow.com/questions/68596666/why-high-level-language-is-considered-slower-than-lower-level>

[9] Sa interneta – Preuzeta i izmenjena definicija mašinskog koda  
<https://www.javatpoint.com/what-is-machine-language>

[10] Sa interneta – Tabela komparacije mašinskog i asemblerskog jezika  
<https://www.geeksforgeeks.org/difference-between-machine-language-and-assembly-language/>