File simulator – kao alat za lakše savladavanje osnova programiranja

Ognjen Bjelica
Laboratorija za embedded sisteme
Elektrotehnički fakultet u Istočnom Sarajevu
Bosna i Hercegovina
ognjen.bjelica@etf.unssa.rs.ba

Sadržaj—**U ovom radu predstavljen je razvoj File simulatora u svrhu lakšeg razumjevanja rada sa fajlovima pri učenju osnova programiranja. Ovaj alat je još uvijek u fazi razvoja, a ovdje je prezentovan dosadašnji progres na razvijanju istog.**

Ključne riječi-file; simulator; C; C#; programiranje;

I. UVOD

Pojam programiranje uglavnom veže samo za razvoj računarskog softvera. Međutim, programiranje se ne veže samo za softver niti samo za računare. Danas je praktično pa moguće isprogramirati svaki uređaj sa kojim dolazimo u dodir, bez obzira na njegovu kompleksnost i način programiranja. Unutar ovog rada focus će biti na programiranju softvera za računare, prvenstveno koristeći programski jezik C, a za razvoj pojedinih alata korišten je programski jezik C#.

Programski jezik C spada u jezike opšte namjene. Razvijen je 1969-1973 god. u AT&T Bell Labs. Svoju popularnost duguje činjenici da je korišten za implementaciju UNIX sistema, a i Linux sistema kasnije. Danas spada među najpopularnije jezike, a veliki broj novih programskih jezika različitih namjena koristi sintaksu koja je slična sintaksi jezika C. Obično se takvi jezici nazivaju „C like languages“ ili „C based languages“ (u slobodnom prevodu jezici slični jeziku C ili jezici zasnovani na jeziku C respektivno) [1], [2]. Neki od danas najpoznatijih jezika zasnovanih na C-u ili njegovoj sintaksi su:

- C++
- C#
- Perl
- Python
- Java
- PHP

Neki od pomenutih jezika su razvijeni i bazirani na C-u, neki samo koriste sličnu sintaksu, ali svi nastoje da privuku što veći broj korisnika (programera) reklamama da su „C like languages“. Aludirajući pri tom da će poznavaocima jezika C, ali i svih sličnih jezika, prelazak na njihov jezik biti jednostavan.

Na Elektrotehničkom fakultetu u Istočnom Sarajevu, pored brojnih drugih pretmeta vezanih za programiranje, na odsjeku zajedničkih osnova (prve dvije godine na svim smjerovima) slušaju se i predmeti Uvod u programiranje i programski jezici koji su prvenstveno fokusirani na savladavanje osnova programiranja u programskom jeziku C [3]. Između ostalog tu se obrađuju sledeće cijeline:

1. Kontrola toka programa (uslovna grananja, petelje, ...)
2. Rad sa jednodimenzionalnim i višedimenzionalnim nizovima
3. Pokazivači i dinamička alokacija memorije
4. Funkcije
5. Složeni tipovi podataka
6. Rad sa fajlovima

Još jednom, ovdje su pobrojane samo određene cjeline (proizvoljno numerisane radi lakšeg referenciranja u nastavku teksta), a kompletni planovi i programi dostupni su na [3]. Od pomenutih cjelina, pokazalo se da studenti najteže savladavaju cjeline 3. i 6. Zbog toga se posljednih godina na vježbama iz ovih predmeta poseban, akcenat stavlja na približavanje ovog gradiva. Tako se za savlađivanje cjeline 3. koriste alati dostupni unutar Microsoft Visual Studio-a za praćenje toka programa i otkrivanje greški, dok se unutar Windows operativnog sistema koristi Task Manager (obično dostupan preko kombinacije tastera Ctrl + Alt + Del ili Ctrl + Shift + Esc) kako bi se na jedostavan način pokazala razlika između statiče i dinamičke alokacije memorije. Pored ovih alata, za sve predviđene primjere crta se raspored, odnosno organizacija memorije (ovo se pokazalo veoma efikasnim i za savladavanje cjeline 4.). Međutim, kada se dođe do cjeline 6. tu jednostavno ne postoji dovoljno dobar način vizuelizacije (pored crtanja na tabli), pa se zbog toga došlo na ideju da se razvije alata za ovu namjenu.

II. RAD SA FAJL SISTEMOM

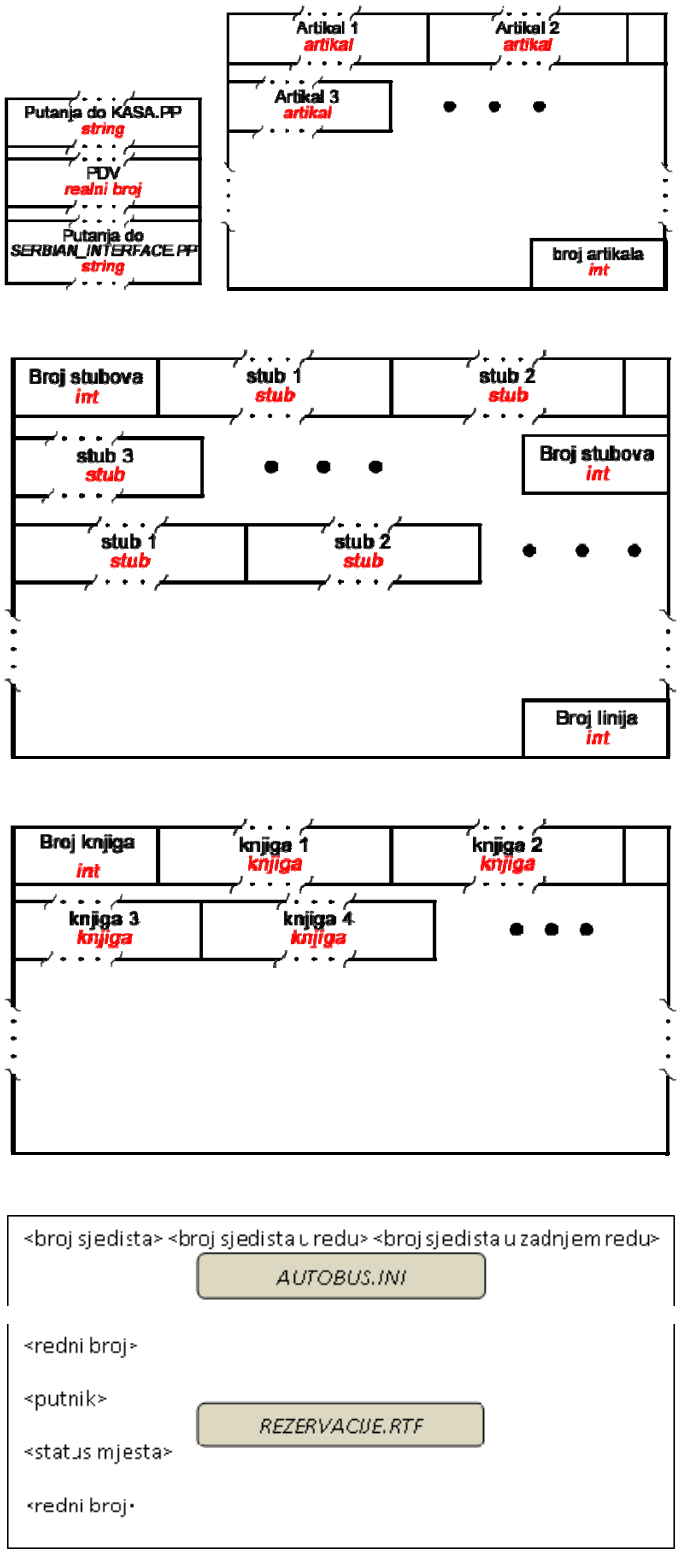
Gradivo koje studenti trebaju da savladaju iz ove oblasti je kratko, ali zahtjeva razumjevanje kako bi se moglo ispravno primjenjivati. Koriste se tekstualni i binarni fajlovi, a gradivo je svedeno na sledeće funkcije:

- fopen – funkcija za otvaranje pokazivača na fajl
- fclose– funkcija za zatvaranje pokazivača na fajl
- fprintf – funkcija za formatiran upis u tekstualni fajl (koristi se radi lakšeg prelaza sa konzole na fajlove)
- fscanf – funkcija za formatirano čitanje iz tekstualnog fajla (koristi se radi lakšeg prelaza sa konzole na fajlove)
- feof – funkcija koja provjera da li se došlo do kraja fajla
- fwrite – funkcija koja se koristi za binarni upis u fajl
- fread – funkcija koja se koristi za binarno čitanje iz fajla
- fseek – funkcija koja se koristi za pozicioniranje unutar fajla
- rewind – kao posebna izvedba prethone funkcije postavlja trenutnu poziciju fajla na početak

Kao što se to može vidjeti, spisak funkcija sveden je na minimum i praktično su ostavljene samo najelementarnije funkcije za rad sa fajlovima. Sama sintaksa ne bi trebala da predstavlja problem, pošto Microsoft Visual Studio ima dobar editor koji nudi pomoć pri kucanju, odnosno programiranju. Neke stvari svi lako savladaju, kao što je to mod otvaranja fajl:

- r – otvaranje prvenstveno radi čitanja (ako fajl postoji, njegov sadržaj neće biti obrisan, ali ako ne postoji otvaranje neće uspjeti i rezultat je NULL)
- w – otvaranje prvenstveno radi upisivanja (ako fajl postoji, njegov sadržaj će biti obrisan, ali ako ne postoji biće kreiran na zadatoj lokaciji (putanja mora biti ispravna - folderi se ne mogu kreirati))
- a – otvaranje radi dopisivanja na kraj fajla (ako fajl postoji, njegov sadržaj neće biti obrisan, ali ako ne postoji biće kreiran na zadatoj lokaciji (putanja mora biti ispravna - folderi se ne mogu kreirati), sve naredbe pozicioniranja se ignorišu)
- + – se koristi kao dodatni znak („r+“, „w+“ i „a+“) kako bi označio da pored osnovne operacije želi i „drugu“. Tako npr. „r+“ označava otvaranje prvenstveno radi čitanja sa mogućnošću upisa.
- b – se koristi kao dodatni znak da označi da je upitanju binarni fajl

i sve njegove konbinacije, svrha i korištenje funkcija, i td [1], [4]. Najveći problem, za pojedine studente, predstavlja razumjevanje pozicije pokazivača unutar fajla. Drugim riječima, kako se ispravno pozicionirati unutar fajla zadate strukture, da se pročita željeni dio ili da se upiše, a bez da se naruši struktura samog fajla. Na Sl. 1 prikazana su četiri primjera organizacije fajla na ispitnim zadacima, kako bi se bolje razumio sam problem.



Slika 1. Primjeri organizacije sadržaja fajlova na ispitnim zadacima iz predmeta Programski jezici.

Primjeri na Sl. 1 preuzeti su direktno iz ispitnih zadataka, dakle i studenti dobiju jasno definisanu strukturu fajla, kao i same fajlove, odnosno funkcije za njihovo generisanje. Uređenje fajlove, odnosno funkcije za generisanje dobijaju, kako na

ispitu ne bi gubili vrijeme na njihovo kreiranje, ali i kako bi mogli preskočiti pojedine dijelove zadatka – da se ne bi desilo da ne znaju jedan dio, pa da dalje ne mogu uraditi ono što bi mogli.

III. ALAT ZA VIZUELIZACIJU – FILE SIMULATOR

Ovaj alat zamišljen je kao dodatna aplikacija koja će korisniku na jednostavan, ali razumljiv način prikazati sadržaj fajla, kao i položaj pokazivača unutar istog. Kao prototip razvijena je aplikacija koja ima implementirano samo čitanje binarnih fajlova – u ovoj fazi bez jednostavnog načina konfiguracije. Unutar korisničkog programa potrebno je uključiti jedan header fajl unutar kog se nalaze funkcije koje komuniciraju sa alatom za vizuelizaciju. Osnovna ideja je da te funkcije imaju što sličnije ime osnovnim funkcijama, kako bi prilikom testiranja aplikacije bile potrebne što manje korekcije koda. Funkcije bi imale prefix „pj_“, dok bi sintaksa ostala potpuno ista. Sledeći primjeri pokazuju kod sa osnovnim funkcijama i kod sa pomoćnim funkcijama za lakše razumjevanje rada sa fajlovima.

```
#include "PJ.h"
void main()
{
    int br_artikala;
    FILE *baza_artikala = "baza.dat";
    fopen(baza_artikala, "r+");
    fseek(baza_artikala, -sizeof(int),
SEEK_END);
    fread(&br_artikala, sizeof(int), 1,
baza_artikala);
    fclose(baza_artikala);
}
```

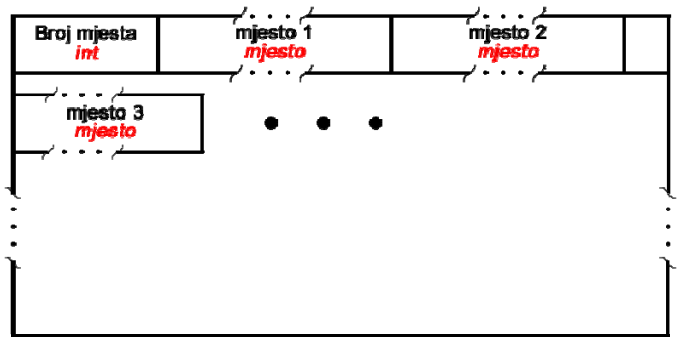
```
#include "PJ.h"
void main()
{
    int br_artikala;
    FILE *baza_artikala = "baza.dat";
    pj_fopen(baza_artikala, "r+");
    pj_fseek(baza_artikala, -sizeof(int),
SEEK_END);
    pj_fread(&br_artikala, sizeof(int), 1,
baza_artikala);
    pj_fclose(baza_artikala);
}
```

Kao što se to iz prethodnih listinga može vidjeti, razlika u kodu je mala. Međutim, bolji efekt se može postići upotrebom predprocesorskih direktiva, npr.:

```
#define FILE_SIMULATOR 1
#if FILE_SIMULATOR == 1
    #define fopen(a,b) pj_fopen(a,b)
    #define fseek(a,b,c) pj_fseek(a,b,c)
    #define fread(a,b,c,c) pj_fread(a,b,c,d)
    #define fclose(a) pj_fclose(a)
#endif
```

Sada korisnik jednom konstantom mjenja režim programa bez potrebe da mjenja ostatak koda. Koja od ovih opcija će biti studentima bliža ostaje da se vidi.

Pogledajmo sada na jednom jestovanom primjeru kako File simulator prikazuje sadržaj jednog jednostavnog fajla. Da bi se primjer razumio, potrebno je prvo prikazati strukturu fajla Sl. 2, zatim definiciju složenog tipa (strukture), kod koji je korisnik ukucao i na kraju prikaz sadržaja fajla Sl. 3.



Slika 2. Organizacija sadržaja fajla koji će se koristiti u nastavku ovog primjera.

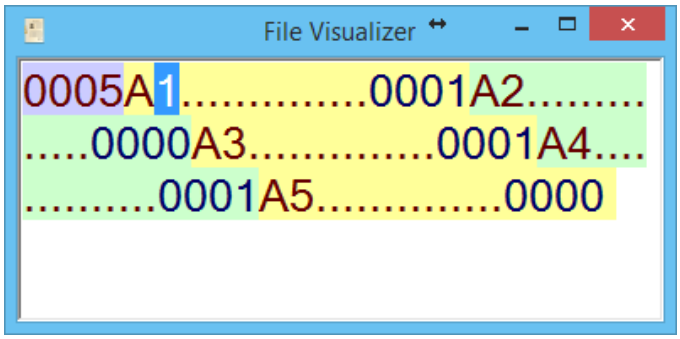
Struktura mjesto definisana je na sledeći način:

```
typedef struct
{
    char oznaka[16];
    int zauzeto;
}mjesto;
```

Ako je korisnik napisao:

```
fseek(parking,          sizeof(int)+sizeof(char),
SEEK_SET);
```

Tada će File simulator prikazati njegov sadržaj u formatu prikazanom na Sl. 3.

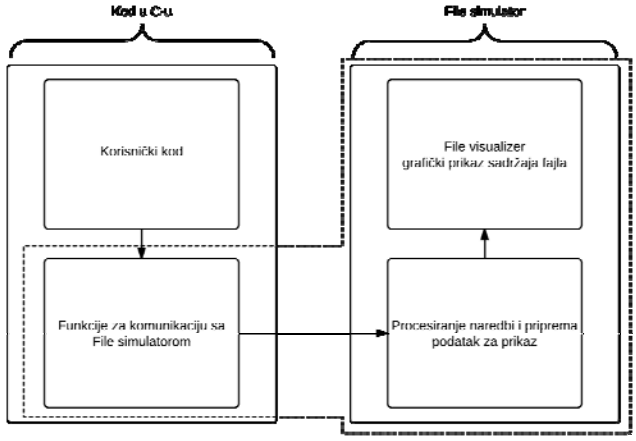


Slika 3. Prikaz sadržaja fajla definisanog u ovom primjeru.

Kao što se to može vidjeti, File simulator je obojio pojedine cjeline. Zbog čega? Da bi studentima na što jedostavniji način razdvojio neke cijeline. Pravila bojenja pojedinih dijelova teksta, su definisana unutar File simulatora. Tako je ovdje prvi broj markiran drugačijom bojom od ostalih cijelina, pošto je definisano da se on javlja samo jednom. Prilikom definisanja pravila bojenja, moguće je za čitavu strukturu primjeniti istu boju, što je na Sl. 3 i prikazano. Radi lakšeg razlikovanja granica strukture, svaka druga struktura ima drugačiju boju

pozadine. Pozicija pokazivača unutar fajla je označena – selektovana, tako da je uočljiva bez obzira na boju pozadine i boju teksta. Iz ovakog prikaza korisnik može lako zaključiti da je pogriješio u pozicioniranju i ispraviti svoju grešku.

File simulator se trenutno sastoji od 3 cjeline, od čega je jedna potpuno odvojena. Blok šema njegove strukture je prikazana na Sl. 4.



Slika 4. Prikaz sadržaja fajla definisanog u ovom primjeru.

Veći dio File simulatora je razvijen u C#, dok su u C-u predviđene samo funkcije koje komuniciraju sa File simulatorom. Pozadinska logika u File simulatoru osluškuje naredbe i vrši pripremu podataka za prikaz. Za prikaz se trenutno koristi Windows Forma sa RichTextBox-om [5].

IV. ZAKLJUČAK

Prikazani alat, čak i u ovoj fazi razvoja, može biti korisno sredstvo u razumjevanju rada sa fajlovima. Međutim, da bi se njegovom potencijal maksimalno iskoristio, potrebno je ipak uložiti dodatno vrijeme u razvoj ovog alata. Kao što je to ranije već pomenuto, prikazani alat predstavlja prototip, čiji je jedini zadatak bio da pokaže da li je moguće ili da li je ne moguće razviti željeni alat. Pri tome se naravno misli na potrebno vrijeme za razvoj.

Dalja napređenja obuhvataju:

- Mogućnost zadavanja konfiguracije fajla (npr. u XML formatu).
- Mogućnost rada sa tekstualnim fajlovima (u trenutnoj verziji implementiran je rad samo sa binarnim fajlovima).
- Mogućnost izvršavanja korak po korak iz samog File simulatora.

Na kraju ostaje samo da se vidi kako će studenti prihvatiti ovaj alat i da li će ga smatrati korisnim sredstvom za savladavanje gradiva.

LITERATURA

[1] Laslo Kraus, *Programski jezik C sa rešenim zadacima*, Deveto dopunjeno izdanje. Beograd: Akademska misao, 2014.

[2] "List of C-based programming languages," *Wikipedia, the free encyclopedia*. 18-Dec-2014.

[3] "Elektrotehnički fakultet, Univerzitet u Istočnom Sarajevu." [Online]. Available: http://www.etf.unssa.rs.ba/. [Accessed: 02-Apr-2015].

[4] "cplusplus." [Online]. Available: http://www.cplusplus.com/reference/. [Accessed: 02-Jun-2015].

[5] "Microsoft Developer Network." [Online]. Available: msdn.microsoft.com. [Accessed: 02-Jun-2015].

ABSTRACT

This paper presents development of File simulator as teaching aid for easier understending of file access in learning basics of programming. This tool is still in development and here is presented its development progress.

File simulator – as teaching aid in learning basics of programming

Ognjen Bjelica