Универзитет у Београду
Факултет организационих наука
Катедра за софтверско инжењерство

ДОКУМЕНТАЦИЈА ЗА ИГРУ „МЕМОРИЈА“

Семинарски рад из Принципа програмирања

Професор:
Душан Савић

Студент:
Иван Баранац, 2023/0391

Београд, 2024.

САДРЖАЈ:

1 УВОД

Игра меморије, такође позната и као "памћење" или "мемо", је класична друштвена игра која је намијењена побољшању когнитивних способности кроз вјежбање краткорочног памћења. Ова игра је једноставна, али изазовна, и погодна је за све узрасте. Основни концепт игре је проналажење парова идентичних картица које су насумично распоређене и окренуте лицем надоле на табли.

Игра меморије има дугу историју и користи се у различитим облицима широм свијета као едукативни алат. Осим што је забава, игра меморије има значајне когнитивне предности. Играчи развијају и унапријеђују своје вештине запажања, концентрације и памћења. Играње ове игре може допринијети бољем извршавању свакодневних задатака који захтијевају брзо и ефикасно обрађивање информација.

Овај пројекат има за циљ да корисницима пружи дигиталну верзију игре „Меморија“, коју могу да играју на својим уређајима. Програм је развијен на програмском језику Ц и намијењен је оперативном систему „Windows“. Користећи библиотеке као што су: stdio.h, stdlib.h, time.h, conio.h, и Windows.h, програм омогућава корисницима да бирају између различитих нивоа тежине и изазивају себе у проналажењу свих парова картица у што мање потеза.

Игра „Меморија“ нуди сљедеће функционалности:

- Избор тежине: Корисници могу да бирају између три нивоа тежине – лако, средње и тешко. Сваки ниво има различити број парова које треба пронаћи.
- Играчко искуство: Интерактивни интерфејс који омогућава унос координата картица које желе да окрену, наравно уз визуелни приказ табле, позадина, музика, све то привлачи кориснике.
- Историја игре: Програм чува податке о броју покушаја и времену потребном за завршетак игре, што даје могућност корисницима да прате свој напредак или се такмиче са другим играчима.

2 КОРИСНИЧКИ ЗАХТЈЕВИ

2.1 Функционални захтјеви

- Програм мора да прикаже уводну поруку која поздравља корисника и тражи његово име.
- Програм треба да омогући кориснику да уђе у главни мени након уноса имена.
- Програм мора да понуди опције за избор нивоа тежине (лако, средње, тешко).
- Корисник мора имати могућност да започне нову игру након избора нивоа тежине.
- Програм треба да иницијализује табелу са паровима картица у складу са изабраном тежином.
- Играч мора бити у могућности да уноси координате како би открио картице.
- Ако корисник унесе неисправне координате, програм треба да прикаже поруку о грешци и затражи поновни унос.
- Ако корисник пронађе пар, програм треба да прикаже поруку о успеху и настави са игром.
- Игра се завршава када корисник пронађе све парове.

Програм мора да чува резултате играња (име играча, број покушаја, ниво тежине) у датотеку.

- Корисник треба да има опцију да прегледа историју својих игара из датотеке.

- Након завршетка игре, програм треба да понуди кориснику опције као што су: играње поново, избор новог нивоа, повратак на главни мени, преглед историје игара и завршетак програма.

2.2 Нефункционални захтјеви

- Програм мора бити интуитиван и лак за коришћење.
- Све корисничке поруке морају бити јасне и информативне.
- Програм мора брзо и ефикасно обрадити унос корисника и ажурирати стање игре.
- Меморијска алокација и ослобађање морају бити оптимално управљани како би се избјегле цурења меморије.
- Програм мора бити компатибилан са Windows оперативним системом.
- Програм мора да садржи провјеру валидности уноса корисника како би се избегле грешке и неправилни уноси.
- Све датотеке за чување и читање података морају бити исправно отворене и затворене.

Као кориснички захтјев се узима у обзир и игрица која се налази на линку:
https://www.igrice123.rs/igrice-memorije/

3 УПОТРЕБА

Када покренете програм, он ће:

1) Прво приказати уводну поруку и затражити име корисника
2) Омогућити избор тежине или бирање нивоа – лако, средње и тешко
3) Покренути игру гдје корисник уноси координате картица које жели да окрене
4) Приказивати резултате уноса координата
5) Када завршимо игру, приказаће укупан број потеза и сваки потез како је одигран
6) Приказати опцију за чување резултата
7) Приказати завршни мени који нуди неколико опција:
    - Играј поново исти ниво
    - Бирање нивоа, наравно, уз чување имена које је унијето на почетку
    - Враћање на почетни мени
    - Историја играња
    - Крај игре
8) Завршити игру

4 ОПИС КОДА

4.1 Функције

- **main**: Покреће програм и иницијализује звучни ефекат и боју конзоле.

```
int main(void) {
    if (!PlaySound(MAKEINTRESOURCE(101),
GetModuleHandle(NULL), SND_RESOURCE | SND_ASYNC | SND_LOOP))
{
        printf("Failed to play sound.\n");
        return 1;
    }
    system("COLOR 2F");
    int ulazak = 0;
    srand(time(NULL));
    predstavljanje(ulazak);
    PlaySound(NULL, 0, 0);
    return 0;
}
```

- **predstavljanje**: Приказује уводну поруку и захтјева име корисника.

```
void predstavljanje(int ulazak) {
    printf("Dobrodosli!!! \n");
    printf("Ovo je igrica - Memorija. Nadji sve parove! :)
\n");
    printf("Vase ime: ");
    char ime[35];
    scanf("%19s", ime);
    ocisti_ekran();
    printf("Zdravo, %s!!!\n", ime);
    ulazak = 1;
    pocetni_meni(ulazak, ime);
}
```

- **ocisti_ekran**: Чисти екран конзоле.

```
void ocisti_ekran() {
    printf("\033[H\033[J");
}
```

- **pocetni_meni**: Приказује мени за избор тежине и покреће одговарајући ниво.

```
void pocetni_meni(int ulazak, char* ime) {
    if (ulazak == 0) {
         predstavljanje(ulazak);
    }
    int redovi, kolone;
    int izbor = biranje_tezine();
    ocisti_ekran();
    switch (izbor)
    {
    case 1:
         redovi = 2;
         kolone = 2;
         printf("Izabrali ste nivo - Lako\nSrecno!\nZa pocetak
igre --> ENTER\n");
         getchar();
         getchar();
         nivo(redovi, kolone, ime, izbor);
         break;
    case 2:
         redovi = 2;
         kolone = 4;
         printf("Izabrali ste nivo - Srednje\nSrecno!\nZa
pocetak igre --> ENTER\n");
         getchar();
         getchar();
         nivo(redovi, kolone, ime, izbor);
         break;
    case 3:
         redovi = 4;
         kolone = 4;
         printf("Izabrali ste nivo - Tesko\nSrecno!\nZa
pocetak igre --> ENTER\n");
         getchar();
         getchar();
         nivo(redovi, kolone, ime, izbor);
         break;
    default:
         printf("Pogresan unos!\n");
         getchar();
         getchar();
         biranje_tezine();
    }
}
```

- **biranje_tezine**: Омогућава кориснику да изабере ниво тежине.

```
int biranje_tezine() {
    printf("Izaberite tezinu:\n1) Lako\n2) Srednje\n3)
Tesko\n");
    int tezina;
    printf("Vas izbor: ");
    scanf("%d", &tezina);
    if (tezina >= 1 && tezina <= 3) {
         return tezina;
    }
    else {
         ocisti_ekran();
         printf("Pogresan unos!\n");
         getchar();
         return biranje_tezine();
    }

}
```

- **nivo**: Покреће игру за одабрани ниво тежине.

```
void nivo(int redovi, int kolone, char* ime, int tezina) {

    FILE* datoteka = fopen("potezi.txt", "w");
    if (datoteka == NULL) {
         printf("Greska pri otvaranju datoteke!");
         return;
    }

    int** mat = (int**)malloc(redovi * sizeof(int*));
    int** otk_mat = (int**)malloc(redovi * sizeof(int*));
    for (int i = 0; i < redovi; i++) {
         mat[i] = (int*)malloc(kolone * sizeof(int));
         otk_mat[i] = (int*)calloc(kolone, sizeof(int));
    }
    int* parovi = (int*)malloc(redovi * kolone * sizeof(int) /
2);

    for (int i = 0; i < redovi * kolone / 2; i++) {
         parovi[i * 2] = i + 1;
         parovi[i * 2 + 1] = i + 1;
    }
    mijesanje(parovi, redovi * kolone);
    ubacivanje_el_matrica(mat, parovi, redovi, kolone);

    int broj_pogodaka = 0;
```

```
    int broj_poteza = 0;
    while (broj_pogodaka < redovi * kolone / 2) {
         char x[10], y[10];
         ocisti_ekran();
         ispis_tabele(mat, otk_mat, redovi, kolone);
         printf("Unesite koordinate prve karte (red, kolona):
");
         scanf("%s%s", x, y);
         int x1 = atoi(x);
         int y1 = atoi(y);

         if (x1 < 1 || y1 < 1 || x1 > redovi || y1 > kolone ||
otk_mat[x1 - 1][y1 - 1])
         {
              printf("Pogresan unos koordinata. Mozete
pokusati ponovo.\n");
              getchar();
              getchar();
              continue;
         }
         otk_mat[x1 - 1][y1 - 1] = 1;

         int brojac = 1;
         int x2, y2;
         while (brojac != 4 && brojac != 0) {
              otk_mat[x1 - 1][y1 - 1] = 1;
              ocisti_ekran();
              ispis_tabele(mat, otk_mat, redovi, kolone);
              printf("Unesite koordinate druge karte (red,
kolona): ");
              scanf("%s%s", x, y);
              brojac++;
              x2 = atoi(x);
              y2 = atoi(y);
              if (brojac == 4) {
                   fprintf(datoteka, "\nPogresne koordinate 3
puta!\n\n");
              }
              if (x2 < 1 || y2 < 1 || x2 > redovi || y2 >
kolone || otk_mat[x2 - 1][y2 - 1] || (x1 == x2) && (y1 ==
y2))
              {
                   printf("Pogresan unos koordinata. Mozete
pokusati ponovo.\n");
                   otk_mat[x1 - 1][y1 - 1] = 0;
                   getchar();
                   getchar();
```

```
                continue;
            }
            brojac = 0;
            otk_mat[x2 - 1][y2 - 1] = 1;

        }

        ocisti_ekran();
        ispis_tabele(mat, otk_mat, redovi, kolone);
        if (brojac < 4) {
            if (mat[x1 - 1][y1 - 1] == mat[x2 - 1][y2 - 1])
{
                fprintf(datoteka, "%d %d\n", x1, y1);
                fprintf(datoteka, "%d %d\n", x2, y2);
                printf("POGODJEN PAR!!!\n");
                fprintf(datoteka, "POGODJEN PAR!!!\n");
                broj_pogodaka++;
            }
            else {
                fprintf(datoteka, "%d %d\n", x1, y1);
                fprintf(datoteka, "%d %d\n", x2, y2);
                printf("POGRESNO - POKUSAJ PONOVO!");
                fprintf(datoteka, "POGRESNO - POKUSAJ
PONOVO!\n");
                otk_mat[x1 - 1][y1 - 1] = 0;
                otk_mat[x2 - 1][y2 - 1] = 0;
            }
            getchar();
            getchar();
        }
        broj_poteza++;

    }
    fclose(datoteka);
    ocisti_ekran();
    printf("Bravo! Pogodio si sve! Rijesio si sve u %d
poteza\n", broj_poteza);
    prikaz_poteza_i_cuvanje_poteza(redovi, kolone, ime,
broj_poteza, tezina, "potezi.txt", "istorija_igranja.txt");
    oslobodi_matricu(mat, redovi);
    oslobodi_matricu(otk_mat, redovi);
    zavrsni_meni(redovi, kolone, ime, broj_poteza, tezina);
}
```

- **mijesanje**: Мијеша картице прије убацивања у матрицу.

```
void mijesanje(int* parovi, int n) {
    for (int i = 0; i < n; i++) {
         int j = rand() % (n);
         int zamjena = parovi[i];
         parovi[i] = parovi[j];
         parovi[j] = zamjena;
    }
}
```

- **ubacivanje_el_matrica**: Убацује картице у матрицу.

```
void ubacivanje_el_matrica(int** mat, int* parovi, int
redovi, int kolone) {
    for (int i = 0; i < redovi; i++) {
         for (int j = 0; j < kolone; j++) {
              mat[i][j] = parovi[i * kolone + j];
         }
    }
}
```

- **ispis_tabele**: Приказује тренутно стање матрице.

```
void ispis_tabele(int** mat, int** otk_mat, int redovi, int
kolone) {
    printf("%c%c%c%c%c", 201, 205, 205, 205, 205);
    for (int i = 0; i < kolone * 3; i++) {
         printf("%c", 205);
    }
    printf("%c\n", 187);

    printf("%c", 186);
    for (int i = 0; i < (kolone * 3 + 4 - strlen("MEMORIJA"))
/ 2; i++) {
         printf(" ");
    }
    printf("MEMORIJA");
    for (int i = 0; i < (kolone * 3 + 4 - strlen("MEMORIJA"))
/ 2; i++) {
         printf(" ");
    }
    printf("%c\n", 186);

    printf("%c%c%c%c%c", 204, 205, 205, 205, 203);
    for (int i = 0; i < kolone * 3; i++) {
         printf("%c", 205);
```

```
    }
    printf("%c\n", 185);

    printf("%c", 186);
    for (int i = -1; i < kolone; i++) {
         if (i >= 0)
              printf(" %d ", i + 1);
         else printf(" %c %c", 254, 186);
    }
    printf("%c\n", 186);

    printf("%c%c%c%c%c", 204, 205, 205, 205, 206);
    for (int i = 0; i < kolone * 3; i++) {
         printf("%c", 205);
    }
    printf("%c\n", 185);

    for (int i = 0; i < redovi; i++) {
         printf("%c %d %c", 186, i + 1, 186);
         for (int j = 0; j < kolone; j++) {
              if (otk_mat[i][j]) {
                   printf(" %d ", mat[i][j]);
              }
              else {
                   printf(" %c ", 'X');

              }

         }
         printf("%c", 186);
         printf("\n");
    }
    printf("%c%c%c%c%c", 200, 205, 205, 205, 202);
    for (int i = 0; i < kolone * 3; i++) {
         printf("%c", 205);
    }
    printf("%c\n", 188);
    printf("\n");
}
```

- **oslobodi_matricu**: Ослобађа меморију заузету матрицом.

```
void oslobodi_matricu(int** mat, int redovi) {
    for (int i = 0; i < redovi; i++) {
         free(mat[i]);
    }
    free(mat);
}
```

- **sacuvaj_u_dat**: Чува резултате у датотеку.

```
void sacuvaj_u_dat(int redovi, int kolone, char* ime, int
broj_poteza, int tezina, const char* naziv_datoteka) {
    FILE* istorija_igranja = fopen(naziv_datoteka, "a");
    time_t t = time(NULL);
    struct tm date = *localtime(&t);
    if (istorija_igranja == NULL) {
         printf("Greska pri otvaranju datoteke!");
         return;
    }
    if (tezina == 1) {

         fprintf(istorija_igranja, "Ova igra je odigrana
%02d/%02d/%d %02d:%02d\n", date.tm_mday, date.tm_mon + 1,
date.tm_year + 1900, date.tm_hour, date.tm_min);
         fprintf(istorija_igranja, "Igrac: %s je u %d poteza
presao nivo - lako!\n\n", ime, broj_poteza);
    }
    else if (tezina == 2) {
         fprintf(istorija_igranja, "Ova igra je odigrana
%02d/%02d/%d %02d:%02d\n", date.tm_mday, date.tm_mon + 1,
date.tm_year + 1900, date.tm_hour, date.tm_min);
         fprintf(istorija_igranja, "Igrac: %s je u %d poteza
presao nivo - srednje!\n\n", ime, broj_poteza);
    }
    else if (tezina == 3) {
         fprintf(istorija_igranja, "Ova igra je odigrana
%02d/%02d/%d %02d:%02d\n", date.tm_mday, date.tm_mon + 1,
date.tm_year + 1900, date.tm_hour, date.tm_min);
         fprintf(istorija_igranja, "Igrac: %s je u %d poteza
presao nivo - tesko!\n\n", ime, broj_poteza);
    }
    fclose(istorija_igranja);
    fclose(istorija_igranja);
    printf("Upisan je rezultat u sistem!");
    getchar();
```

```
    ocisti_ekran();
}
```

- **procitaj_iz_dat**: Чита историју играња из датотеке.

```
void procitaj_iz_dat(int redovi, int kolone, char* ime, int
broj_poteza, int tezina, char* naziv_datoteka, int poziv) {
    FILE* datoteka = fopen(naziv_datoteka, "r");
    if (datoteka == NULL) {
         printf("Greska pri otvaranju datoteke!");
         return;
    }
    printf("Ovo je istorija igranja:\n");
    char karakter;
    while ((karakter = fgetc(datoteka)) != EOF) {
         printf("%c", karakter);
    }

    fclose(datoteka);
    if (poziv == 1) {
         getchar();
         ocisti_ekran();
         zavrsni_meni(redovi, kolone, ime, broj_poteza,
tezina);
    }
}
void predstavljanje(int ulazak) {
    printf("Dobrodosli!!! \n");
    printf("Ovo je igrica - Memorija. Nadji sve parove! :)
\n");
    printf("Vase ime: ");
    char ime[35];
    scanf("%19s", ime);
    ocisti_ekran();
    printf("Zdravo, %s!!!\n", ime);
    ulazak = 1;
    pocetni_meni(ulazak, ime);
}
```

- **prikaz_poteza_i_cuvanje_poteza**: Приказује потезе и омогућава чување резултата.

```
void prikaz_poteza_i_cuvanje_poteza(int redovi, int kolone,
char* ime, int broj_poteza, int tezina, const char*
istorija_igranja, const char* potezi) {
    procitaj_iz_dat(redovi, kolone, ime, broj_poteza, tezina,
"potezi.txt", 0);
    printf("\nDa li zelite da sacuvate rezultat?\n1) Da, hocu!
*_*\n2) Ne, mogu ja to bolje!\n");
    printf("Vas izbor: ");
    int izbor;
    scanf("%d", &izbor);
    getchar();
    if (izbor == 1) {
         cuvanje_poteza();
         sacuvaj_u_dat(redovi, kolone, ime, broj_poteza,
tezina, "istorija_igranja.txt");
    }
    else if (izbor == 2) {
         ocisti_ekran();
         printf("Ti to mozes!");
         getchar();
    }
    else {
         ocisti_ekran();
         printf("Pogresan unos! Mozete pokusati ponovo.\n");
         prikaz_poteza_i_cuvanje_poteza(redovi, kolone, ime,
broj_poteza, tezina, "potezi.txt", "istorija_igranja.txt");
    }
    ocisti_ekran();
}
```

- **zavrsni_meni**: Приказује мени након завршене игре.

```
void zavrsni_meni(int redovi, int kolone, char* ime, int
broj_poteza, int tezina) {
    printf("Izaberite opciju:\n1) Igraj ponovo!\n2) Biranje
nivoa\n3) Vrati na pocetni meni\n4) Istorija igranja\n5) Kraj
igre\n");
    int opcija;
    printf("Vas izbor: ");
    scanf("%d", &opcija);
    getchar();
    if (opcija == 1) {
         ocisti_ekran();
         nivo(redovi, kolone, ime, tezina);
    }
    else if (opcija == 2) {
         ocisti_ekran();
         pocetni_meni(1, ime);
    }
    else if (opcija == 3) {
         ocisti_ekran();
         predstavljanje(0);
    }
    else if (opcija == 4) {
         ocisti_ekran();
         procitaj_iz_dat(redovi, kolone, ime, broj_poteza,
tezina, "istorija_igranja.txt", 1);
    }
    else if (opcija == 5) {
         ocisti_ekran();
         printf("Hvala sto ste igrali nasu igru!\n");
    }
    else {
         ocisti_ekran();
         printf("Pogresan unos!\n");
         zavrsni_meni(redovi, kolone, ime, broj_poteza,
tezina);
    }
}
```

5 ЗАКЉУЧАК

Овај програм је написан у Ц језику, тако да је обезбијеђено брзо извршавање и ефикасност. Међутим, наравно, да постоји простор за даљу оптимизацију и развој. Неки од могућих праваца за побољшање које сам уочио су:

- Додавање тајмера
- Подршка за више играча, било локално или путем мреже, може много повећати интерактивност и привлачност игре
- Напредне статистике, нпр. просјечно вријеме за прелазак игрице....
- Додатни нивои и тежине, као и можда другачија правила за неке тежине
- Додавање графичког корисничког интерфејса

6 ЛИТЕРАТУРА

- **Dr Bojana Dimić Surla, dr Dragan Urošević (2020).** Uvod u programiranje sa primerima u programskom jeziku C. RAF (Računarski fakultet).
- **Kernighan, B. W., & Ritchie, D. M. (1988).** *The C Programming Language* (2nd Edition). Prentice Hall.
- **Harbison, S. P., & Steele, G. L. (2002).** *C: A Reference Manual* (5th Edition). Prentice Hall.
- **King, K. N. (2008).** *C Programming: A Modern Approach* (2nd Edition). W. W. Norton & Company.
- **Schildt, H. (2000).** *C: The Complete Reference* (4th Edition). McGraw-Hill.
- **Prata, S. (2013).** *C Primer Plus* (6th Edition). Addison-Wesley.
- **Van der Linden, P. (1994).** *Expert C Programming: Deep C Secrets*. Prentice Hall.
- **Kochan, S. G. (2004).** *Programming in C* (3rd Edition). Sams Publishing.