REPUBLIKA E SHQIPËRISË

UNIVERSITETI I TIRANËS

FAKULTETI I SHKENCAVE TË NATYRËS

DEGA: INFORMATIKË

Projekt për Sisteme Logjike

Tema: *"Ndërtimi i diagramës dhe tabelës së gjëndjeve të sistemeve sinkrone"*

Punoi:

Lorela Elezaj

Malvin Abazi

Marin Dulja

Sara Tanku

Valentina Furxhi

Pranoi:

Alba Çomo

Abstract

Programi është shkruar në gjuhën C++, me GUI të ndërtuar me OpenGL. Ai merr të dhëna nga përdoruesi me anë të një skedari dhe krijon një seri gjendjesh, të cilat përdoren për të gjetur hyrjen e marre si input në një sekuence binare.

Përmbajtja

- Leximi i inputit
- Ndertimi i gjendjeve
- Kërkimi për hyrjen në sekuencën binare
- Afishimi i tabelës së gjendjeve
- Afishimi i diagramës së gjendjeve

Hyrje

Programi që u ndërtua ka si synim të krijojë diagramën dhe tabelën e gjendjeve sinkrone. Kjo mundësohet duke marrë si input nga përdoruesi një sekuence binare dhe hyrje, dhe në fund të japi daljen 1 sa herë hyrja gjendet në sekuencën binare. Më pas do të duhet të afishojë diagramën e gjëndjeve për këtë sekuencë, si dhe tabelën e gjendjeve.

Leximi i Inputit

Leximi i inputit bëhet me anë të një skedari inupt te quajtur "Te Dhena.txt". Ajo është e formatuar si më poshtë:

Përdoruesi i programit fut sipas dëshirës hyrjen dhe sekuencën binare. Në këtë dokument do të ilustrohet përdorimi I programit me hyrjen 1001 dhe sekuencën binare 011100101100110, si në screenshot-in e mësipërm.

```
ifstream input("Te Dhena.txt", ios::in);

int main(int argc, char **argv) {

    if (!input.is_open()) {                    // kontrollojme nqs e gjejme/hapim file-in e inputit
        cout << "File-i nuk u hap!";
        exit(1);
    }

    else {                                     // u hap file-i

        string in_line;                        // stringu ku do mbajme hyrjen nga file-i rresht pas rreshti

        cout << "\n\t Duke lexuar te dhenat...";
        cout << "\n ----------------------------------------";

        int rreshti = 1;                       // mban rreshtin ku jemi duke lexuar aktualisht ne file
        while (input >> in_line) {             // kontrollon nqs jemi ne fund te file-it dhe njekohesisht merr rreshtin e rradhes
            if (rreshti == 2) {                // dmth jemi te rreshti 2
                lexoNeVektor(in_line, hyrje);  // hyrja ruhet shifer pas shifre ne nje vektor
                s_hyrje = in_line;             // hyrja ruhet dhe e tera ne nje string
            }
            else if (rreshti == 5) {           // dmth jemi te rreshti 5
                lexoNeVektor(in_line, sekBinare);
            }
            rreshti++;
        }                                      // mbaron leximi
```

Hapim skedarin me një ***ifstream*** të quajtur ***input***. Kontrollojmë nëse skedari u hap me sukses, dhe në qoftë se jo I japim fund programit. Nëse u hap saktë, lexojmë inputin rresht pas rreshti me anë të një cikli *while*. Kodi brënda kushtit të *while* njëkohësisht kontrollon nëse ka ende rreshta për tu lexuar në skedar, si dhe merr rreshtin e tanishëm. Numrin e rreshtit e ruajmë në variablin ***int rreshti***. Kur merr vlerën 2 dhe 5 (përkatësisht rreshtat ku kemi hyrjen dhe sekuencën binare), i ruajmë shifër pas shifre në një vektor. Hyrjen e ruajmë gjithashtu si *string* për lehtësi në veprime në një pjesë të mëposhtme të programit.

```
    if (hyrje.size() == 0) {
        cout << "\n\n Hyrja nuk mund te jete bosh! Rregulloni hyrjen dhe provoni perseri. \n\n";
        return 0;
    }
    if (sekBinare.size() == 0) {
        cout << "\n\n Sekuenca binare nuk mund te jete bosh! Rregulloni sekuencen dhe provoni perseri. \n\n";
        return 0;
    }
    for (int i = 0; i < hyrje.size(); i++) {    // kontrollojme gjithe shifrat e hyrjes per input te padeshiruar
        if (hyrje[i] != 1 && hyrje[i] != 0) {   // nqs shifra nuk eshte 0 ose 1
            cout << "\n\n Hyrja permban shifra qe nuk jane 0/1. Rregulloni hyrjen dhe provoni perseri. \n\n";
            return 0;
        }
    }
    for (int i = 0; i < sekBinare.size(); i++) {
        if (sekBinare[i] != 1 && sekBinare[i] != 0) { // nqs shifra nuk eshte 0 ose 1
            cout << "\n\n Sekuenca binare permban shifra qe nuk jane 0/1. Rregulloni sekuencen dhe provoni perseri. \n\n ";
            return 0;
        }
    }
    cout << "  Inputi u lexua me sukses! \n";
} // end leximin e file-it te inputit
```

Pasi është marrë inputi, verifikojmë që është marrë i saktë. Nëse hyrja ose sekuenca binare janë bosh, ose përmbajnë shifra/karaktere jo 0/1, përdoruesit I kthehet një mesazh errori dhe programi mbyllet. Përndryshe, afishohet mesazhi *"Inputi u lexua me sukses!"* dhe vazhdon pjesa tjetër e programit.

Ndërtimi i Gjendjeve

Për ndërtimin dhe ruajtjen e të dhënave të gjendjeve të ndryshme u ndërtua një klasë ***Gjendje***. Në skedarin *"Gjendje.h"* ruhen variablat e instancës dhe deklarimet e metodave dhe në *"Gjendje.cpp"* implementohen.

Variablat e instancës:

```
private:
    Gjendje * pas[2];       // gjendjet pasardhese te kalimeve me 0 dhe 1
    int rezultat[2];        // rezultatet e kalimeve
    char emer;              // emri i gjendjes, psh A, B, etj.
    std::string sekuence;   // sekuenca e inputeve qe na kane cuar te kjo gjendje
```

Klasa përmban metodat get/set për secilën nga variablat e instancës, si dhe një metodë që I afishon të gjitha variablat.

Ndërkohë ne skedarin *"Main.cpp"* krijojme shënjusa drejt gjendjes se fundit ku sekuenca ishte '0' dhe '1' (përkatësisht ***Gjendje *fundit0 dhe *fundit1***) si dhe një drejt gjendjes së fundit të krijuar, ***Gjendje *gjFundit***. Gjendjet e krijuara ruhen në nje array.

Si fillim krijojme gjendjen e pare me emër ***'A'***. Sekuenca e saj do të jetë sa e kundërta e shifrës së parë të hyrjes të marrë si input. Kalimi më atë shifër e le tek vetja dhe kalimi me shifrën e pare të hyrjes e kalon të gjendja e rradhes, të cilës I vëmë emrin ***'B'.*** Në rastin tonë, ****fundit0*** do të inicializohet më gjendjen ***A*** dhe ****fundit1*** me ***B***

```
                                    // ndertimi i gjendjes se pare
if (hyrje[0] == 0) { ... }
else {                              // shifra e pare ishte 1

    gj[0].setPasardhes(0, gj[0]);   // kalimi me 0 na le tek vetja
    gj[0].setPasardhes(1, gj[1]);   // kalimi me 1 na con tek gjendja e rradhes, B

    gj[0].setRezultat(0, 0);        // te gjitha kalimet japin rezultat 0 ne kete moment
    gj[0].setRezultat(1, 0);

    gj[0].setSekuence("0");         // per kete gjendje
    gj[1].setSekuence("1");         // per gjendjen e rradhes

    fundit0 = &gj[0];               // dmth gjendja A
    fundit1 = &gj[1];               // dmth gjendja B
}
gj[0].setEmer('A');                 // vendosi emrin A
```

Këtu tregohet se rasti yn, ku shifra e parë e hyrjes ishte 1, kapet nga rasti else.

Më pas me anë të një cikli *for* bredhim të gjithë shifrat e hyrjes së kërkuar. Për cdo gjendje kalimi me këtë shifër e con te gjendja pasardhëse, ndërsa kalimi me të kundërtën e saj e con përkatësisht tek ***fundit0*** dhe ***fundit1***. Për aq kohë sa nuk jemi në fund te hyrjes, të gjitha kalimet do të kenë rezultat 0. Sekuenca e gjendjes krijohet duke marrë sekuencën e gjendjes së mëparshme dhe duke I konkatinuar shifrën e rradhës së hryjes. Pasi ka mbaruar cikli, të gjitha gjendjet përvecse të fundit do të jenë të lidhura me njera-tjetrën, dhe të gjithë variablat e instancës së tyre do të kenë vlera.

Për te caktuar kalimet e gjendjes së fundit marrim sekuencën e saj dhe I shtojme njëhere 0, pastaj 1. Me anë të një cikli marrim substring të kësaj nga shkronja e I-te deri në fund, dhe e krahasojmë këtë substring me sekuencat e secilës gjendje. Pasardhësi I gjendjes së fundit me 0/1 është ajo gjendje qe ka sekuencë që përputhet me substringun në fjalë. Në qoftë së dalim nga cikli dhe nuk u gjet asnjë përputhje, pasardhësit e kalimit me 0/1 do jenë përkatësisht ***fundit0*** dhe ***fundit1.***

```
                                        // ndertimi i kalimeve te gjendjes se fundit
string s = gjFundit->getSekuence();
s.append("0");                          // do shofim ku do shkoje me kalim me 0
for (int i = 1; i < H_GJATESI; i++) {   // fillon nga i qe ne substring te mos kapim shifren e pare te sekuences por nga e dyta deri ne fund
    bool found = false;
    string sub_sekuence = s.substr(i);  // substring i tere sekuences se gjendjes se fundit, nga shifra e i-te deri ne fund

    for (int j = H_GJATESI + 1; j >= 0; j--) {  // kerko mbrapsht ne vektorin e gjendjeve derisa gjen nje me sekuence si e jona
        if (gj[j].getSekuence() == sub_sekuence) {
            found = true;
            gjFundit->setPasardhes(0, gj[j]);   // meqe sekuenca eshte njelloj, do kalojme me 0 ketu
            if (sub_sekuence == s_hyrje)        // nqs substringu i sekuences plus 0'ne qe i beme append eshte sa hyrja, do te thote se rezultati i kalimit do jete 1
                gjFundit->setRezultat(0, 1);
            else
                gjFundit->setRezultat(0, 0);    // perndryshe rezultati do jete 0
            break;
        }
    }

    if (found)
        break;

    if (i == H_GJATESI - 1 && !found) {         // nqs jemi ne fund dhe s'kemi gjetur gje
        gjFundit->setPasardhes(0, *fundit0);    // do te thote se kalimi me 0 do na coj te vendi i fndit ku sekuenca ishte vtm 0
        gjFundit->setRezultat(0, 0);
    }
}
```

Këtu tregohet kodi për kalimin me 0, por procesi është i njejtë edhe për kalimin me 1.

Me këtë hap përfundon lidhja e gjithë gjendjeve me njera tjetrën.

Gjetja e Hyrjes ne Sekuencën Binare

Për të gjetur pozicionin e hyrjes në sekuencën binare të dhënë nga përdoruesi mjafton të futemi në një cikël *for* ku kapim të gjithë shifrat e sekuencës binare një nga një. Fillojmë me nje iterator te quajtur ***gjTani***, i cili shenjon tek gjendja e parë, ajo që quajtëm '***A***'. Kur shifra e sekuencës është 0 kalojmë tek pasardhësi i kalimit me 0 i ***gjTani***, dhe kur është 1 kalojmë tek pasardhësi me 1.

Kur ***gjTani*** shënjon tek ***gjFundit*** do të thotë se kemi kaluar në tërë gjendjet dhe gjetur hyrjen. Në këtë moment afishojmë daljen **'1'**, sipas kërkesave të projektit, si dhe pozicionin ku u gjet hyrja.

```
                                        // gjejme vendndodhjen e hyrjes ne sekuencen binare qe morem si input
Gjendje *gjTani = &gj[0];               // do bredhim neper gjendje duke filluar nga e para, dmth gjendja A
for (int i = 0; i < S_GJATESI; i++) {
    if (sekBinare[i] == 0)
        gjTani = gjTani->getPasardhes(0);
    else                                // dmth sekBinare[i] == 1
        gjTani = gjTani->getPasardhes(1);

    if (gjTani == gjFundit) {           // dmth jemi ne fund te hyrjes
        cout << "\n  Pozicioni i hyrjes: " << i - hyrje.size() << " -> "; // afishojme pozicionin e shifres se pare te hyrjes
        cout << 1;
    }
}
```

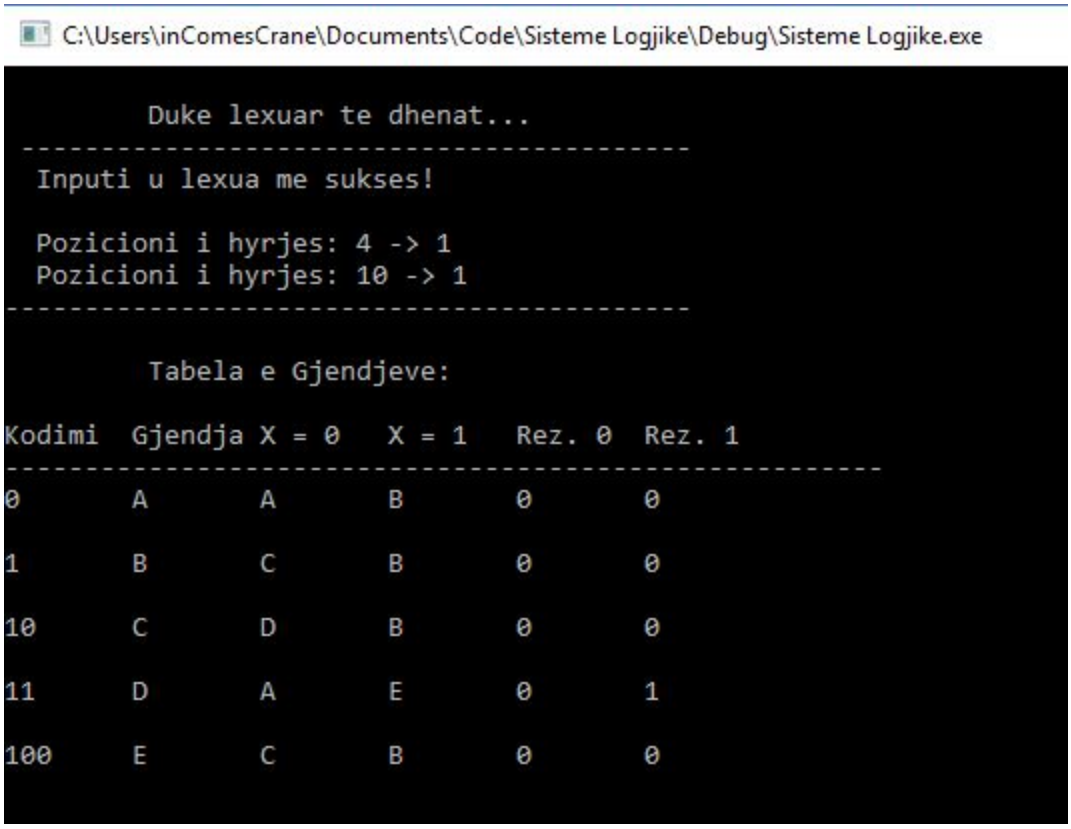
Afishimi i tabelës së gjendjeve

Për të ndërtuar tabelën e gjendjeve bredhim nëpër *array* ku kemi të ruajtura me rradhë gjendjet dhe afishojmë vlerat e variablave të tyre:

```
cout << "\n-------------------------------------------\n";;
cout << "\n\t Tabela e Gjendjeve: \n\n";
cout << "Kodimi\t" << "Gjendja\t" << "X = 0\t" << "X = 1\t" << "Rez. 0\t" << "Rez. 1\n";
cout << "-----------------------------------------------------\n";
for (int i = 0; i < H_GJATESI + 1; i++) {

    int rezultat_0 = gj[i].getRezultat(0); // kap vlerat e rezulatateve
    int rezultat_1 = gj[i].getRezultat(1);
    cout << konvertoBinar(i) << "\t" << emraGjendje[i] << "\t"
        << gj[i].getPasardhes(0)->getEmer() << "\t" << gj[i].getPasardhes(1)->getEmer() << "\t"
        << rezultat_0 << "\t" << rezultat_1 << endl << endl;
}
```

Rezultati i gjithë afishimeve të programit në console:

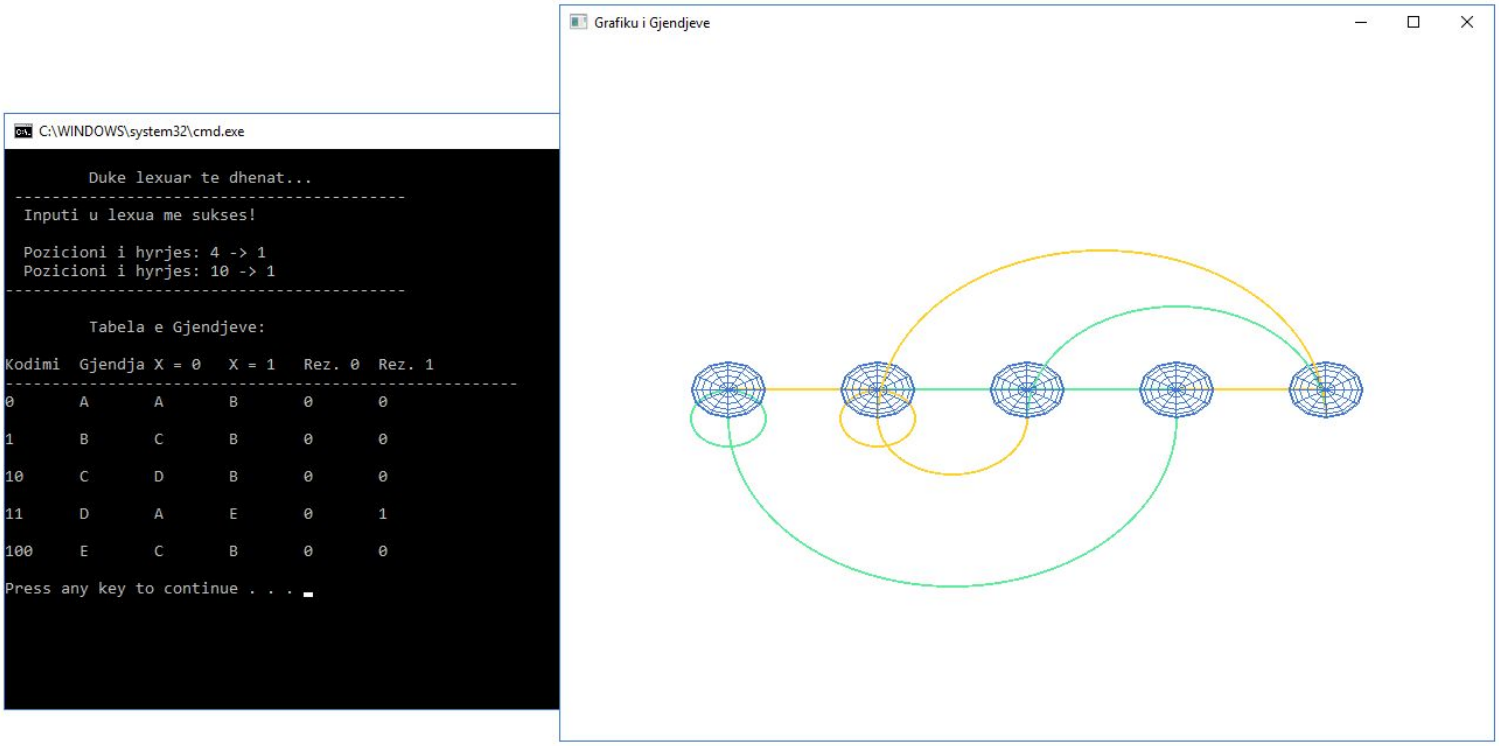


Afishimi i diagramës së gjendjeve

Diagrama e gjendjeve afishohet në një dritare me anë te kodit OpenGL, me paraqitje të tillë:

- Gjendjet - Sfera blu
- Kalimet para (psh. nga gjendja **B** tek **D**) - Vija te drejta
- Kalimet mbrapa (psh. nga gjendja **C** tek **A**) - Vija te harkuara
- Kalimet që të lënë tek e njejta gjendje - Rrathë

Për kalimet me 0 vijat kanë ngjyrë jeshile dhe për kalimet me 1 kanë ngjyrë të verdhë. Tabela e gjendjeve ndihmon gjithashtu për leximin e diagramës



Përfundim

Programi arrin të gjejë hyrje deri në gjashtë shifra tek një sekuencë binare të dhënë, si dhe të gjejë hyrje në cfarëdolloj pozicioni, edhe nëse mbivendoset disa herë në sekuencë (si psh. gjetja e hyrjes *‘1001’* dy herë në sekuencën *‘1001001’*).

Referenca

C++ Reference, www.cplusplus.com/reference/

OpenGL® 4.5 Reference Pages, www.khronos.org/registry/OpenGL-Refpages/gl4/