Laboratorinis darbas su pirštų antspaudų skaitytuvu Upek Eikon

Darbo tikslas

Susipažinti pirštų antspaudų skaitytuvo funkcionalumu ir išmokti naudotis programinių funkcijų biblioteka

Naudojamos priemonės

1. Upek Eikon pirštų antspaudų skaitytuvas
2. Upek Eikon C++ programinių funkcijų biblioteka http://www.personalas.ktu.lt/~darbirv/sauga/BSAPI.pdf
3. Microsoft Visual C++ 2010

Teorinė dalis

Pirštų antspaudų skaitytuvai gali būti įvairių tipų: optiniai, terminiai, talpiniai, ultragarsiniai, slėginiai. Upek Eikon pirštų antspaudų skaitytuvo jutikliai yra talpiniai. Talpinės varžos jutiklio elementai matuoja talpumą: ten, kur yra piršto griovelis (oras), talpumas mažesnis, o tose vietose, kurias liečia oda, – didesnis. Jutiklio elementai išdėstyti siauroje juostelėje, dėl to piršto antspaudo nuskaitymui reikalingas piršto perbraukimas.

Pagrindinės skaitytuvo funkcijos yra nuskaityti piršto antspaudą ir palyginti su anksčiau sukurtu specialiu piršto antspaudo šablonu. Šablone yra užkoduotos piršto antspaudo savybės – tam tikri taškai, kuriuose išsiskiria ar baigiasi linijos (šie taškai dažnai vadinami detalėmis (angl. minutiae) arba Galtono savybėmis). Iš šablono atkurti piršto antspaudo negalima dėl saugumo sumetimų, nes, bent jau teoriškai, piršto antspaudu gali pasinaudoti kiti asmenys, ir tada antspaudo savininkas netenka svarbaus savęs identifikavimo būdo.

Prieš pradedant naudotis Upek Eikon pirštų antspaudų skaitytuvu, reikia inicializuoti funkcijų biblioteką ir susikurti naują sesiją su įrenginiu. Antspaudo šablono sukūrimui reikalinga ilgesnė operacija – tris, keturis kartus pilnai perbraukti pirštu per sensorių. Naujai nuskaityto antspaudo palyginimui su šablonu užtenka perbraukti ir vieną kartą.

Upek Eikon pirštų antspaudų skaitytuvas turi ir papildomas funkcijas, nesusijusias su antspaudų nuskaitymu: gali dirbti navigacijos režime kaip „biometrinė pelė“ (pvz. valdyti žymeklį ekrane). Taip pat yra šviesos diodas, kurio švytėjimo dažnį ir intervalą galima įvairiai keisti, taip siunčiant vartotojui įvairius signalus.

Instrukcija

Paleisti Microsoft Visual C++ 2010 programavimo, atidaryti pateiktą C++ projektą Eikon.vcxproj ir pagal individualią užduotį panaudoti reikiamas pirštų antspaudų skaitytuvo bibliotekos funkcijas.

1. Upek Eikon skaitytuvo inicializavimo funkcijos

1.1. ABSInitialize

```
ABS_STATUS ABSInitialize(
     void
)
```

Inicializuoja BSAPI.DLL biblioteką. Ši funkcija turi būti iškviečiama prieš kviečiant kitas funkcijas. Grąžina `ABS_STATUS`. Reikšmė `ABS_STATUS_OK (0)` reiškia, kad biblioteką inicializuoti pavyko.

1.2. ABSTerminate

```
ABS_STATUS ABSTerminate (
     void
)
```

Uždaryti BSAPI.DLL biblioteką. Ši funkcija turi būti iškviečiama uždarius visas aktyvias sesijas su įrenginiu. Grąžina `ABS_STATUS`. Reikšmė `ABS_STATUS_OK (0)` reiškia, kad biblioteką uždaryti pavyko.

1.3. ABSOpen

```
ABS_STATUS ABSOpen(
     IN const ABS_CHAR *pszDsn
     OUT ABS_CONNECTION *phConnection
)
```

Sukurti naują sesiją su įrenginiu. Naudojami du parametrai:

- **`*pszDsn`** parametras naudojamas nurodyti, prie kurio įrenginio jungtis, pvz. „`usb`“. Jei yra keletas USB įrenginių, tai reikia nurodyti konkretų įrenginį įrenginys, pvz. „`usb,device=\\?\usb#vid_0483&pid_2016#5&20890ddc&0&1#{d5620e51-8478-44bd-867e-aac02f883a00}`“ (ABSEnumerateDevices funkcija gali grąžinti sąrašą įrenginių).
- **`*phConnection`** parametras grąžina nuorodą į naują sesiją, kurioje bus vykdomos kitos operacijos su įrenginiu.

Funkcija grąžina `ABS_STATUS`. Reikšmė `ABS_STATUS_OK (0)` reiškia, kad naują sesiją sukurti pavyko.

1.4. ABSClose

```
ABS_STATUS ABSClose(
     IN ABS_CONNECTION hConnection
)
```

Uždaryti sukurtą sesiją su įrenginiu. Naudojamas vienas parametras:

- **`*hConnection`** – nuorodą į aktyvią sesiją, kuria norima uždaryti.

Funkcija grąžina `ABS_STATUS`. Reikšmė `ABS_STATUS_OK (0)` reiškia, kad sesiją uždaryti pavyko.

1.5. ABSEnumerateDevices

```
ABS_STATUS ABSEnumerateDevices(
     IN const ABS_CHAR *pszEnumDsn
     OUT ABS_DEVICE_LIST **ppDeviceList
)
```

Gauti prijungtų antspaudų skaitytuvų sąrašą. Naudojami du parametrai:

- **`*pszEnumDsn`** parametras naudojamas nurodyti ryšio sąsają, pvz. „`usb`“.
- **`**ppDeviceList`** parametras grąžina nuorodą į įrenginių sąrašą (**`dev_list->NumDevices`** – įrenginių kiekis, **`dev_list->List[i].DsnSubString`** – nuoroda į konkretų įrenginį (naudojama ABSOpen() funkcijoje))

Funkcija grąžina `ABS_STATUS`. Reikšmė `ABS_STATUS_OK (0)` reiškia, kad operacija įvykdyta sėkmingai.

2. Upek Eikon skaitytuvo biometrinės funkcijos

2.1. ABSEnroll

```
ABS_STATUS ABSEnroll(
     IN ABS_CONNECTION hConnection
     IN ABS_OPERATION *pOperation
     OUT ABS_BIR **ppEnrolledTemplate
     IN ABS_DWORD dwFlags
)
```

Sukurti naują piršto antspaudo šabloną. Tam reikia keletą kartų nuskenuoti piršto antspaudą. Naudojami keturi parametrai:

- **`*hConnection`** - nuoroda į aktyvią sesiją, kurioje bus vykdoma operacija.
- **`*pOperation`** - vykdomos operacijos parametrai (žiūr. `ABS_OPERATION`).
- **`**ppEnrolledTemplate`** - nuoroda į naujai sukurtą antspaudo šabloną.
- **`dwFlags`** - parametras rezervuotas ateičiai, naudoti 0.

Funkcija grąžina `ABS_STATUS`. Reikšmė `ABS_STATUS_OK (0)` reiškia, kad naują šabloną sukurti pavyko.

2.2. ABSVerify

```
ABS_STATUS ABSVerify(
     IN ABS_CONNECTION hConnection
     IN ABS_OPERATION *pOperation
     IN ABS_DWORD dwTemplateCount
     IN ABS_BIR **pTemplateArray
     OUT ABS_LONG *pResult
     IN ABS_DWORD dwFlags
)
```

Paimti naują antspaudą, sukurti iš jo šabloną, patikrinti ar toks šablonas jau yra masyve `pTemplateArray` ir, jei yra, sugrąžinti sąrašo numerį. Naudojami šeši parametrai:

- **`*hConnection`** - nuoroda į aktyvią sesiją, kurioje bus vykdoma operacija.

- **`*pOperation`** – vykdomos operacijos parametrai (žiūr. `ABS_OPERATION`).
- **`dwTemplateCount`** – elementų kiekis šablonų masyve `pTemplateArray`.
- **`**pTemplateArray`** – nuoroda į šablonų masyvą.
- **`*pResult`** – gražinamas masyvo elemento numeris, jei antspaudų šablonai sutapo, arba -1, jei nėra nė vieno sutampančio šablono.
- **`dwFlags`** – naudoti 0. Parametras naudojamas keisti funkcijos elgseną tam tikrais atvejais (žiūr. BSAPI dokumentaciją).

Funkcija grąžina `ABS_STATUS`. Reikšmė `ABS_STATUS_OK (0)` reiškia, kad operacija įvykdyta sėkmingai.

2.3. ABSVerifyMatch

```
ABS_STATUS ABSVerifyMatch(
     IN ABS_CONNECTION hConnection
     IN ABS_BIR *pEnrolledTemplate
     IN ABS_BIR *pVerificationTemplate
     OUT ABS_BOOL *pResult
     IN ABS_DWORD dwFlags
)
```

Patikrinti ar du pirštų antspaudų šablonai sutampa tarpusavyje. Naudojami penki parametrai:

- **`*hConnection`** – nuoroda į aktyvią sesiją, kurioje bus vykdoma operacija.
- **`*pEnrolledTemplate`** – pirmas šablonas palyginimui.
- **`*pVerificationTemplate`** – antras šablonas palyginimui.
- **`*pResult`** – gražinamas palyginimo rezultatas: `ABS_TRUE`, jei šablonai sutampa, `ABS_FALSE`, jei nesutampa.
- **`dwFlags`** – parametras rezervuotas ateičiai, naudoti 0.

Funkcija grąžina `ABS_STATUS`. Reikšmė `ABS_STATUS_OK (0)` reiškia, kad operacija įvykdyta sėkmingai.

3. Kitos Upek Eikon skaitytuvo funkcijos

3.1. ABSGetLastErrorInfo

```
void ABSGetLastErrorInfo(
OUT ABS_DWORD *pErrorCode
OUT const ABS_CHAR **ppErrorMessage
```

Grąžinti informaciją apie klaidą. Naudojami du parametrai:

- **`*pErrorCode`** – klaidos kodas.
- **`**ppErrorMessage`** – klaidos pranešimas.

3.2. ABSNavigate

```
ABS_STATUS ABSNavigate(
     IN ABS_CONNECTION hConnection
     IN ABS_OPERATION *pOperation
     IN ABS_DWORD dwFlags
)
```

Perjungti skaitytuvą į navigacijos (arba „biometrinės pelės“) darbo režimą. Naudojami trys parametrai:

- **`*hConnection`** – nuoroda į aktyvią sesiją, kurioje bus vykdoma operacija.

- **`*pOperation`** – vykdomos operacijos parametrai (žiūr. `ABS_OPERATION`).
- **`dwFlags`** – parametras rezervuotas ateičiai, naudoti 0.

Funkcija negrąžina `ABS_STATUS_OK (0)` reikšmės, tol kol operacija nenutraukiama naudojant funkciją `ABSCancelOperation()` arba kol neįvyksta klaida.

3.3. ABSCancelOperation

```
ABS_STATUS ABSCancelOperation(
     IN ABS_CONNECTION hConnection
     IN ABS_DWORD dwOperationID
)
```

Nutraukti skaitytuvo vykdomą operaciją. Naudojami du parametrai:

- **`*hConnection`** – nuoroda į aktyvią sesiją, kurioje vykdoma operacija.
- **`dwOperationID`** – vykdomos operacijos identifikatorius arba 0 – nutraukti aktyvią operaciją (žiūr. `ABS_OPERATION`).

Funkcija grąžina `ABS_STATUS`. Reikšmė `ABS_STATUS_OK (0)` reiškia, kad operacija įvykdyta sėkmingai.

3.4. ABSSetLED

```
ABS_STATUS ABSSetLED(
     IN ABS_CONNECTION hConnection
     IN ABS_DWORD dwMode
     IN ABS_DWORD dwLED1
     IN ABS_DWORD dwLED2
)
```

Valdyti šviesos diodų švytėjimo dažnį ir intervalą. Naudojami keturi parametrai:

- **`*hConnection`** – nuoroda į aktyvią sesiją, kurioje vykdoma operacija.
- **`dwMode`** – diodo valdymo režimas: `ABS_LED_MODE_MANUAL` – diodų valdymas per parametrus `dwLED1` ir `dwLED2`, `ABS_LED_MODE_AUTO` – diodų valdymą atiduoti BSAPI, `ABS_LED_MODE_OFF` – išjungti diodus.
- **`dwLED1`** – parametras pirmo diodo valdymui. Parametre nusirodo mirgėjimo šablonas ir šablono elemento švytėjimo laikas. 0-15 bitai apibrėžia mirgėjimo šabloną (1 – diodas šviečia, 0 – nešviečia), 16-19 bitai – kiek laiko skirta vienam iš šablono bitui (1 – 1 milisekundė, 2 – 2 ms, 3 – 4 ms, ... , 15 – 16384 ms).
- **`dwLED2`** – parametras antro diodo valdymui (jei toks yra, Upek Eikon turi tik vieną diodą).

Funkcija grąžina `ABS_STATUS`. Reikšmė `ABS_STATUS_OK (0)` reiškia, kad operacija įvykdyta sėkmingai.

3.5. ABSSetGlobalParameter

```
ABS_STATUS ABSSetGlobalParameter(
     IN ABS_DWORD dwParamID
     IN ABS_DATA *pParamValue
)
```

Nustatyti globalius parametrus, kurie įtakoja BSAPI funkcijų veikimą. Naudojami du parametrai:

- **`dwParamID`** – parametro identifikatorius (žr. dokumentaciją ABS_PARAM_xxxx, ..).
- **`*pParamValue`** – parametro reikšmė (žr. dokumentaciją ABS_PARAM_xxxx, ..).

4. BSAPI bibliotekos kintamųjų tipai

4.1. Paprasti tipai

```
typedef char ABS_CHAR Signed integer type (1 byte)

typedef unsigned char ABS_BYTE Unsigned integer type (1 byte)

typedef short ABS_SHORT Signed integer type (2 bytes)

typedef unsigned short ABS_WORD Unsigned integer type (2 bytes)

typedef int ABS_LONG Signed integer type (4 bytes)

typedef unsigned int ABS_DWORD Unsigned integer type (4 bytes)

typedef int ABS_BOOL Boolean value (zero, non-zero)

typedef ABS_LONG ABS_STATUS

typedef ABS_DWORD ABS_CONNECTION
```

4.2. Specifiniai tipai

ABS_BIR

```
typedef struct abs_bir {
   ABS_BIR_HEADER Header;
   ABS_BYTE Data[ABS_VARLEN];
} ABS_BIR
```

Saugoti piršto antspaudo šabloną (angl. Biometric Identification Record). Susideda iš dviejų elementų:

- **`Header`** – antraštė (žiūr. `ABS_BIR_HEADER`).
- **`Data[ABS_VARLEN]`** – piršto antspaudo šablono duomenys.

ABS_DEVICE_LIST

```
typedef struct abs_device_list {
   ABS_DWORD NumDevices;
   ABS_DEVICE_LIST_ITEM List[ABS_VARLEN];
} ABS_DEVICE_LIST
```

Naudojamas saugoti įrenginių sąrašą. Susideda iš dviejų elementų:

- **`NumDevices`** – įrenginių kiekis.
- **`List[ABS_VARLEN]`** – įrenginių sąrašas.

ABS_DEVICE_LIST_ITEM

```
typedef struct abs_device_list_item {
   ABS_CHAR DsnSubString[260];
   ABS_BYTE reserved[256];
} ABS_DEVICE_LIST_ITEM
```

Naudojamas saugoti įrenginių sąrašo elementui. Susideda iš dviejų elementų:

- **`DsnSubString` –** įrenginių įrenginį identifikuojanti eilutė, naudojama funkcijoje ABSOpen.
- **`reserved` –** nenaudojama (rezervuota ateičiai).

ABS_NAVIGATION_DATA

```
typedef struct abs_navigation_data {
   ABS_LONG DeltaX;
   ABS_LONG DeltaY;
   ABS_BOOL FingerPresent;
} ABS_NAVIGATION_DATA
```

Saugoti papildomiems duomenims naudoti interakcijai navigacijos režime. Susideda iš trijų elementų:

- **`DeltaX` –** horizontalios koordinatės pokytis.
- **`DeltaY` –** vertikalios koordinatės pokytis.
- **`FingerPresent` –** ar pirštas ant sensoriaus, ABS_TRUE – taip, ABS_FALSE – ne.

ABS_OPERATION

```
typedef struct abs_operation {
   ABS_DWORD OperationID;
   void* Context;
   ABS_CALLBACK Callback;
   ABS_LONG Timeout;
   ABS_DWORD Flags;
} ABS_OPERATION
```

Saugo operacijos duomenis. Susideda iš penkių elementų:

- **`OperationID` –** operacijos identifikatorius.
- **`Context` –** bet kokia vartotojo sukurta nuoroda, perduodama operacijos vykdymui.
- **`Callback` –** nuoroda į funkciją, kuri gražina informaciją iš vykdomos operacijos, pvz. kokia operacijos eiga, ką vartotojas turi padaryti ir pan.
- **`Timeout` –** kiek ilgai laukti vartotojo veiksmo.
- **`Flags` –** operacijos darbo režimo nustatymas (žiūr. ABS_OPERATION_FLAG_xxxx).

4.3. Globalus parametrai

ABS_PARAM_CONSOLIDATION_COUNT_MIN – minimalus piršto perbraukimo kiekis antspaudo šablonui sukurti. Galimos reikšmės nuo 1 iki 10. Pagal nutylėjimą reikšmė yra 3.

ABS_PARAM_CONSOLIDATION_COUNT_MAX – maksimalus piršto perbraukimo kiekis antspaudo šablonui sukurti. Galimos reikšmės nuo 1 iki 10. Pagal nutylėjimą reikšmė yra 10.

ABS_PARAM_CONSOLIDATION_TYPE – šablono kūrimo operacijos tipas. Galimos reškšmės:
ABS_CONSOLIDATION_NORMAL (0) – normalus (šablonas kuriamas iš kelių arba vieno geriausio antspaudo)
ABS_CONSOLIDATION_CONVENIENT (1) – laisvas (laisvesnis tipas už normalų)
ABS_CONSOLIDATION_STRICT (2) – griežtas (visi antspaudai turi būti vieno piršto)

ABS_PARAM_MATCH_LEVEL – saugumo lygis lyginant du šablonus su funkcija ABSVerifyMatch(). Pagal nutylėjimą naudojama ABS_MATCH_MEDIUM_SECURITY. Galimos reikšmės:
ABS_MATCH_MIN_SECURITY (1), ABS_MATCH_LOWER_SECURITY (2), ABS_MATCH_MEDIUM_SECURITY (3), ABS_MATCH_HIGHER_SECURITY (4), ABS_MATCH_MAX_SECURITY (5).

5. BSAPI bibliotekos funkcijų panaudojimo pavyzdžiai

5.1. Operacijos „Callback“ funkcija

```
static void BSAPI
callback(const ABS_OPERATION* p_operation, ABS_DWORD msg, void* data)
{
    UNREFERENCED_PARAMETER(p_operation);

    switch(msg) {
        case ABS_MSG_PROCESS_BEGIN:
        case ABS_MSG_PROCESS_END:
            break;

        case ABS_MSG_PROCESS_SUSPEND:
            printf("   operation has been suspended\n");
            break;
        case ABS_MSG_PROCESS_RESUME:
            printf("   operation has been resumed\n");
            break;

        case ABS_MSG_PROCESS_PROGRESS:
        {
            ABS_PROCESS_PROGRESS_DATA* progress_data =
                                (ABS_PROCESS_PROGRESS_DATA*) data;
            if(progress_data->Percentage <= 100) {
                printf("   operation in progress (%d%%)...\n",
                                        (int)progress_data->Percentage);
            } else {
                printf("   operation in progress...\n");
            }
            break;
        }
        case ABS_MSG_PROCESS_SUCCESS:
            printf("   success\n");
            break;
        case ABS_MSG_PROCESS_FAILURE:
            printf("   failure\n");
            break;

        case ABS_MSG_PROMPT_SCAN:
            printf("   swipe the finger\n");
            break;
        case ABS_MSG_PROMPT_TOUCH:
            printf("   touch the sensor\n");
            break;
        case ABS_MSG_PROMPT_KEEP:
            printf("   keep finger on the sensor\n");
            break;
        case ABS_MSG_PROMPT_LIFT:
            printf("   lift your finger away from the sensor\n");
            break;
        case ABS_MSG_PROMPT_CLEAN:
            printf("   clean the sensor\n");
            break;

        case ABS_MSG_QUALITY_CENTER_HARDER:
            printf("   bad quality: center and harder\n");
            break;
        case ABS_MSG_QUALITY_CENTER:
            printf("   bad quality: center\n");
            break;
        case ABS_MSG_QUALITY_TOO_LEFT:
            printf("   bad quality: too left\n");
            break;
        case ABS_MSG_QUALITY_TOO_RIGHT:
            printf("   bad quality: too right\n");
            break;
        case ABS_MSG_QUALITY_HARDER:
            printf("   bad quality: harder\n");
            break;
        case ABS_MSG_QUALITY_TOO_LIGHT:
            printf("   bad quality: too light\n");
            break;
        case ABS_MSG_QUALITY_TOO_DRY:
```

```
            printf("   bad quality: too dry\n");
            break;
        case ABS_MSG_QUALITY_TOO_SMALL:
            printf("   bad quality: too small\n");
            break;
        case ABS_MSG_QUALITY_TOO_SHORT:
            printf("   bad quality: too short\n");
            break;
        case ABS_MSG_QUALITY_TOO_HIGH:
            printf("   bad quality: too high\n");
            break;
        case ABS_MSG_QUALITY_TOO_LOW:
            printf("   bad quality: too low\n");
            break;
        case ABS_MSG_QUALITY_TOO_FAST:
            printf("   bad quality: too fast\n");
            break;
        case ABS_MSG_QUALITY_TOO_SKEWED:
            printf("   bad quality: too skewed\n");
            break;
        case ABS_MSG_QUALITY_TOO_DARK:
            printf("   bad quality: too dark\n");
            break;
        case ABS_MSG_QUALITY_BACKWARD:
            printf("   bad quality: backward movement detected\n");
            break;
        case ABS_MSG_QUALITY_JOINT:
            printf("   bad quality: joint detected\n");
            break;

        case ABS_MSG_NAVIGATE_CHANGE:
        case ABS_MSG_NAVIGATE_CLICK:
            break;

        case ABS_MSG_DLG_SHOW:
        case ABS_MSG_DLG_HIDE:
            break;

        case ABS_MSG_IDLE:
            break;
    }
}
```

5.2. Operacija

```
static ABS_OPERATION op = {
    0,
    NULL,
    60000,
    ABS_OPERATION_FLAG_LL_CALLBACK
};
```

5.3. Klaidos duomenys

```
ABS_STATUS status;
ABS_DWORD code;
const ABS_CHAR* message;

ABSGetLastErrorInfo(&code, &message);
printf("   status: %ld\n", (long)status);
printf("   code:   %ld\n", (long)code);
printf("   message: '%s'\n", message);
```

5.4. Sukurti naują sesiją

```
ABS_STATUS res;

//Sukuriam nauja sesija su skaitytuvu
res=ABSOpen("usb", &conn);

if(res == ABS_STATUS_OK){
       printf("Sukurem nauja sesija");
} else {
       klaida(res);
}
```

5.5. Sukurti naują sesiją, pasirenkant konkretų įrenginį

```
ABS_STATUS res;
ABS_DEVICE_LIST* dev_list;

res = ABSEnumerateDevices("usb", &dev_list);

/* Sukuriam nauja sesija su pasirinktu skaitytuvu */
printf("Jungiames prie '%s'...\n", dev_list->List[dev_index].DsnSubString);
res = ABSOpen(dev_list->List[dev_index].DsnSubString, &conn);
```

5.6. Sukurti naują piršto antspaudo šabloną

```
res = ABSEnroll(conn, &op, &tset[slot], 0);

if(res == ABS_STATUS_OK) {
       printf("Naujas antspaudo sablonas itrauktas i saraso elementa Nr.%d\n", slot);
} else {
       klaida(res);
}
```

5.7. Palyginti naują piršto antspaudą su pasirinktu šablonu

```
ABS_LONG matching_slot;

res = ABSVerify(conn, &op, 1, &tset[slot], &matching_slot, 0);
if(matching_slot == 0) {
       printf("Sutampa\n");
} else if(matching_slot < 0) {
       printf("Nesutampa\n");
}
```

5.8. Patikrinti, ar naujas piršto antspaudas yra šablonų sąraše

```
ABS_BIR* tmp_tset[TSET_SIZE];
int tmp_slot[TSET_SIZE];

/* Sukurti nauja laikina sablonu masyva */
for(i = 0; i < TSET_SIZE; i++) {
    if(tset[i] != NULL) {
        tmp_tset[count] = tset[i];
        tmp_slot[count] = i;
        count++;
    }
}

/* Tikrinam ar anspaudas yra sarase */
res = ABSVerify(conn, &op, count, tmp_tset, &index, 0);

if(index >= 0) {
    printf("Antspaudas yra saraso elemente Nr.%d\n", tmp_slot[index]);
} else {
    printf("Antspaudo sarase nera\n");
}
```

5.9. Palyginti antspaudų šablonus tarpusavyje

```
res = ABSVerifyMatch(conn, tset[slot1], tset[slot2], &match, 0);

if(match)
    printf("Antspaudai %d ir %d sutampa.\n", slot1, slot2);
else
    printf("Antspaudai %d ir %d nesutampa.\n", slot1, slot2);
```

5.10. Išsaugoti antspaudo šabloną į failą

```
FILE *f;

f = fopen(templateFileName, "wb");
if (f == NULL) {
    printf("Failo atidarymo klaida '%s'.\n", templateFileName);
    return;
}

if (fwrite(tset[slot], tset[slot]->Header.Length, 1, f) != 1) {
    printf("Negalima irasyti %u baitu i faila '%s'.\n",
        tset[slot]->Header.Length, templateFileName);
```

```
    fclose(f);
    return;
}
```

5.11. Užkrauti antspaudo šabloną iš failo

```
f = fopen(templateFileName, "rb");

tset[slot] = (ABS_BIR*)malloc(fileSize);
if (tset[slot] == NULL) {
    printf("Negaliu issiskirti %ld baitu atminties\n", fileSize);
    fclose(f);
    return;
}

if (fread(tset[slot], fileSize, 1, f) != 1) {
    printf("Negaliu nuskaityti %ld baitu is failo '%s'\n", fileSize,
        templateFileName);
    free(tset[slot]);
    tset[slot] = NULL;
    fclose(f);
    return;
}
```

5.12. Valdyti skaitytuvo LED

```
res = ABSSetLED(conn, ABS_LED_MODE_MANUAL, 0x400FF, 0);
if(res != ABS_STATUS_OK) {
    klaida(res);
    return;
}
```

5.13. Nustatyti globalius parametrus

```
ABS_STATUS res;
ABS_DATA *p_value = new ABS_DATA;

p_value->Length=1;
p_value->Data[0]=5;
res = ABSSetGlobalParameter(ABS_PARAM_CONSOLIDATION_COUNT_MIN,p_value);

p_value->Length=1;
p_value->Data[0]=ABS_CONSOLIDATION_STRICT;
res = ABSSetGlobalParameter(ABS_PARAM_CONSOLIDATION_TYPE,p_value);

p_value->Length=1;
p_value->Data[0]=ABS_MATCH_MAX_SECURITY;
res = ABSSetGlobalParameter(ABS_PARAM_MATCH_LEVEL,p_value);
if(res != ABS_STATUS_OK) {
    klaida(res);
    return;
}
```

6. Laboratorinio darbo užduočių pavyzdžiai

1. Sukurti keletą antspaudų šablonų, surašyti į masyvą, nuskaityti naują antspaudą ir patikrinti, ar yra masyve
2. Sukurti keletą antspaudų šablonų, surašyti į masyvą, palyginti antspaudų šablonus tarpusavyje, keičiant globalų parametrą ABS_PARAM_MATCH_LEVEL
3. Sukurti keletą antspaudų šablonų, surašyti į masyvą, patikrinti, ar naujas atspaudas sutampa su konkrečiu elementu iš masyvo
4. Sukurti keletą antspaudų šablonų, surašyti į masyvą, pagal skirtingą piršto antspaudą įvykdyti komandą
5. Sukurti to paties piršto keletą šablonų ir eksportuoti į failą, palyginti failus tarpusavyje (skiriasi ar ne)
6. Nuskaityti ir parodyti ekrane piršto antspaudą (ne šabloną)
7. Panaudoti skaitytuvo navigacijos funkciją
8. Sugalvoti savo užduotį