```c
#include <sdl.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <math.h>
#include "sirvid.h"

void saaPyhadArr(void);
void print_sirvi(SDL_Surface *ekraan, kuupaev k);
void lisaPilt(SDL_Surface *ekraan, char strAsukoht[], int x, int y);
void saa_nPaevadArr(void);
void lisaPilt2(SDL_Surface *ekraan, char strAsukoht[], int x, int y);
int lisaInt(int nr, int intPikkus, SDL_Surface *img, int x, int y);
void lisaPilt_hw(SDL_Surface *ekraan, char strAsukoht[], int x, int y, int *w, int *h);
void lisaKuupaevad(kuupaev k1, kuupaev k2, SDL_Surface *img, int x, int y);
void saa_kFaasidArr(void);
void saa_lPyhadArr(void);
void looSirvi(kuupaev k);
void suhtleja(void);
kuupaev saaKuupaev(kuupaev alg_k);
void prindiSirviPyha(kuupaev k);

Uint32 getpixel(SDL_Surface *surface, int x, int y);
void putpixel(SDL_Surface *surface, int x, int y, Uint32 pixel);


int pPyhadArr[26][3];
int nPaevadArr[8][2];
int kFaasidArr[8][3];
int lPyhadArr[9][3];

int main(int argc, char *argv[])
{
    printf("**********************\n");
    printf("Sirvide Generaator 1.0\n");
    printf("**********************\n\n");

    saaPyhadArr();
    saa_nPaevadArr();
    saa_kFaasidArr();
    saa_lPyhadArr();

    suhtleja();
    printf("Programm sulgeb!\n");
    return(0);
}
void suhtleja(void)
{
    kuupaev k = saaHetkeneKuupaev();
    int intSuhtlus = 1, onSirvi = 0;
    char strVastus[20], *uusRida;

    prindiKuupaev("Tänane kuupaev on: ", k);
    prindiKuupaev("Sirvi kalendri järgi: ", gregoorius_sirviks(k));

    do
```

```
    {
        switch(intSuhtlus)
        {
            case 1:
                printf("\nKas sa soovid näha sirvi kalendrit tänasest kuupäevast?\n");
                printf("[j][jah] - Jah\n[e][ei]  - Ei\n\n");
                break;
            case 2:
                printf("\n");
                printf("Et näha kõiki võimalike käske siseta:   [k][kasud]\n");
                printf("Et väljuda programmist sisesta:         [x][lahku]\n\n");
                intSuhtlus = 3;
                break;
            case 3:
                break;
            case 4:
                printf("\n");
                printf("[k][kasud]            - Näitab kõiki süsteemis olevaid käske\n");
                printf("[s][sirvi]            - Käivitab sirvi kalendri generaatori,\n");
                printf("                        kuupäevaga, mis on mälus\n");
                printf("[sk][sisesta kuupaev] - Laseb kasutajal sisestada uue kuupäeva
mällu\n");
                printf("[x][lahku]            - Sulgeb programmi\n");
                printf("[p][kuupaev]          - Prindi kuupäaev mälus\n");
                printf("[ps][sirvi kuupaev]   - Prindi kuupäaev mälus, Sirvi kalendri
järgi\n");
                printf("[hk][tana kuupaevaks] - Sea mälus olev kuupaev tänase kuupäevaga\n");
                printf("\n");
                break;
        }

        printf("Sisesta Käsk:\n>>");
        if(fgets(strVastus, 20, stdin))
        {
            uusRida = strchr(strVastus, '\n');
            if(uusRida != NULL) *uusRida = '\0';


            if(strcmp(strVastus, "lahku") == 0 || strcmp(strVastus, "x") == 0) intSuhtlus = 0;
            else if(strcmp(strVastus, "kasud") == 0 || strcmp(strVastus, "k") == 0)
intSuhtlus = 4;
            else if(strcmp(strVastus, "sirvi") == 0 || strcmp(strVastus, "s") == 0)
            {
                looSirvi(k);
                intSuhtlus = 3;
            }
            else if(strcmp(strVastus, "kuupaev") == 0 || strcmp(strVastus, "p") == 0)
                prindiKuupaev("Mälus olev kuupaev on: ", k);
            else if(strcmp(strVastus, "sirvi kuupaev") == 0 || strcmp(strVastus, "ps") == 0)
            {
                prindiKuupaev("Mälus olev kuupaev Sirvi kalendri järgi on: ",
gregoorius_sirviks(k));
                prindiSirviPyha(k);
                intSuhtlus = 3;
            }
            else if(strcmp(strVastus, "tana kuupaevaks") == 0 || strcmp(strVastus, "hk") == 0
)
```

```
            {
                k = saaHetkeneKuupaev();
                prindiKuupaev("Mälus olev kuupaev on: ", k);
            }
            else if(strcmp(strVastus, "sisesta kuupaev") == 0 || strcmp(strVastus, "sk") == 0)
            {
                k = saaKuupaev(k);
                intSuhtlus = 3;
            }
            else if(intSuhtlus == 1 && (strcmp(strVastus, "jah") == 0 || strcmp(strVastus,
"j") == 0))
            {
                looSirvi(k);
                intSuhtlus = 2;
            }
            else if(intSuhtlus == 1 && (strcmp(strVastus, "ei") == 0 || strcmp(strVastus, "e"
) == 0)) intSuhtlus = 2;
            else
            {
                printf("Teadmatu käsk, palun sisesta uus.\n");
                intSuhtlus = 2;
            }
            if(intSuhtlus == 1) intSuhtlus = 2;

        }else
            {
                printf("Teadmatu Viga!\n");
                intSuhtlus = 0;
            }

    }while(intSuhtlus != 0);
}
void prindiSirviPyha(kuupaev k)
{
    int onTeada, intLiikuvPyha = onSirviLiikuvPyha(k, &onTeada);

    switch(intLiikuvPyha)
    {
        case 1: printf("Kevadine Pööripaev\n"); break;
        case 2: printf("Suvine Pööripaev\n"); break;
        case 3: printf("Sügisene Pööripaev\n"); break;
        case 4: printf("Munapyha\n"); break;
        case 5: printf("Maahengaus\n"); break;
        case 6: printf("Suvisted\n"); break;
        case 7: printf("Urbepäev\n"); break;
        case 8: printf("Kihlakud\n"); break;
        case 9: printf("Eidepäev\n"); break;
    }
    if(!onTeada && intLiikuvPyha > 0) printf("NB! - Täpsed pööripäevad aasta %d kohta ei ole
teada\n", k.aasta);

    switch(onSirviPysivPyha(k))
    {
        case 1: printf("Korjusepäev\n"); break;
        case 2: printf("Taliharjapäev\n"); break;
        case 3: printf("Pudrupäev\n"); break;
        case 4: printf("Luuvalupäev\n"); break;
```

```
        case 5: printf("Sirgupäev \n"); break;
        case 6: printf("Marjapunapäev\n"); break;
        case 7: printf("Kynnipäev\n"); break;
        case 8: printf("Karjalaskepäev\n"); break;
        case 9: printf("Ligupäev\n"); break;
        case 10: printf("Leedopäev\n"); break;
        case 11: printf("Heinaleedo\n"); break;
        case 12: printf("Karusepäev\n"); break;
        case 13: printf("Jakepäev\n"); break;
        case 14: printf("Esimene rukkiema päev\n"); break;
        case 15: printf("Suur rukkiema päev\n"); break;
        case 16: printf("Viimane rukkiema päev\n"); break;
        case 17: printf("Ussi urgu minemise päev\n"); break;
        case 18: printf("Kasupäev\n"); break;
        case 19: printf("Kolletamispäev\n"); break;
        case 20: printf("Hingepäev\n"); break;
        case 21: printf("Mardipäev\n"); break;
        case 22: printf("Lambapäev\n"); break;
        case 23: printf("Tõnnipäev\n"); break;
        case 24: printf("Ajastaja päev\n"); break;
        case 25: printf("Jõulud\n"); break;
        case 26: printf("Pekopäev\n"); break;
    }
}
kuupaev saaKuupaev(kuupaev alg_k)
{
    kuupaev k, ret = alg_k;
    char strVastus[20], *uusRida;
    printf("Sisesta uus kuupäev formaadiga dd.mm.yyyy:\n>>");
    if(fgets(strVastus, 20, stdin))
    {
        uusRida = strchr(strVastus, '\n');
        if(uusRida != NULL) *uusRida = '\0';

        strVastus[2] = '\0';
        strVastus[5] = '\0';

        k.paev = atoi(strVastus);
        k.kuu = atoi(strVastus+3);
        k.aasta = atoi(strVastus+6);

        prindiKuupaev("Sisestatud Kuupäev on: ", k);
        if(onKuupaev(k)) ret = k;
        else
            printf("\nAntud kuupaev ei ole õige formaadiga!\nKuupaeva ei sisestatud
mällu!\n\n");
    }else
        printf("Süsteemi Viga! - Kuupaeva sisestud ebaõnnestus!");

    return(ret);
}
void looSirvi(kuupaev k)
{
    if(SDL_Init(SDL_INIT_VIDEO) < 0)
    {
        printf("SDL'i käivitu: %s\n", SDL_GetError());
        exit(1);
```

```
    }

    SDL_WM_SetCaption("Sirvide Generaator 1.0", NULL);

    SDL_Surface *ekraan;
    ekraan = SDL_SetVideoMode(640,480,32,SDL_HWSURFACE|SDL_DOUBLEBUF);
    if(ekraan == NULL)
    {
        printf("Ei ole võimeline käivitama 640x480 video: %s\n", SDL_GetError());
        exit(1);
    }
    int intValmis = 0;

    print_sirvi(ekraan, k);

    clock_t c1, c2;
    int onAll = 0, onParemale = 0, onVasakule = 0;

    while(!intValmis)
    {
        SDL_Event asi;
        while(SDL_PollEvent(&asi))
        {
            if(asi.type == SDL_QUIT) intValmis = 1;
            if(asi.type == SDL_KEYDOWN)
            {
                if(asi.key.keysym.sym == SDLK_ESCAPE) intValmis = 1;
                if(asi.key.keysym.sym == SDLK_LEFT)
                {
                    k = saaKuupaev_lisaArv(k, -1.0);
                    print_sirvi(ekraan, k);
                    onVasakule = 1;
                    onAll = 1;
                    c1 = clock();
                }
                if(asi.key.keysym.sym == SDLK_RIGHT)
                {
                    k = saaKuupaev_lisaArv(k, 1.0);
                    print_sirvi(ekraan, k);
                    onParemale = 1;
                    onAll = 1;
                    c1 = clock();
                }
            }
            if(asi.type == SDL_KEYUP)
            {
                onAll = 0;
                onParemale = 0;
                onVasakule = 0;
            }
        }

        if(onAll)
        {
            c2 = clock();
            if(c2 - c1 > 500)
            {
```

```
                if(onVasakule)
                {
                    k = saaKuupaev_lisaArv(k, -1.0);
                    print_sirvi(ekraan, k);
                }
                if(onParemale)
                {
                    k = saaKuupaev_lisaArv(k, 1.0);
                    print_sirvi(ekraan, k);
                }
            }
        }

    }

    SDL_Quit();
}
void saa_nPaevadArr(void)
{
    FILE *f;
    int i = 1;
    f = fopen("txt/n.txt", "r");
    if(f != NULL)
    {
        fscanf(f, "%d", &nPaevadArr[0][0]);
        nPaevadArr[0][1] = nPaevadArr[0][0];
        while(!feof(f))
        {
            fscanf(f, "%d", &nPaevadArr[i][0]);
            fscanf(f, "%d", &nPaevadArr[i][1]);
            i++;
        }
        fclose(f);
    }
}
void saaPyhadArr(void)
{
    FILE *f;
    int i = 0;
    f = fopen("txt/p.txt", "r");
    if(f != NULL)
    {
        while(!feof(f))
        {
            fscanf(f, "%d", &pPyhadArr[i][0]);
            fscanf(f, "%d", &pPyhadArr[i][1]);
            fscanf(f, "%d", &pPyhadArr[i][2]);
            i++;
        }
        fclose(f);
    }
}
void saa_lPyhadArr(void)
{
    FILE *f;
    int i = 0;
    f = fopen("txt/L.txt", "r");
```

```
    if(f != NULL)
    {
        while(!feof(f))
        {
            fscanf(f, "%d", &lPyhadArr[i][0]);
            fscanf(f, "%d", &lPyhadArr[i][1]);
            fscanf(f, "%d", &lPyhadArr[i][2]);
            i++;
        }
        fclose(f);
    }
}
void saa_kFaasidArr(void)
{
    FILE *f;
    int i = 0;
    f = fopen("txt/k.txt", "r");
    if(f != NULL)
    {
        while(!feof(f))
        {
            fscanf(f, "%d", &kFaasidArr[i][0]);
            fscanf(f, "%d", &kFaasidArr[i][1]);
            fscanf(f, "%d", &kFaasidArr[i][2]);
            i++;
        }
        fclose(f);
    }
}
void print_sirvi(SDL_Surface *ekraan, kuupaev k)
{
    int i, intPaev, intLaius = 0, intSirviH = 330,
        intPyha, intAbi, onJoulud = 0, onKihlakud = 0,
        arv_KuuLoomine, arv_Taiskuu, h, w,
        onTaiskuu = 0, onKuuLoomine = 0, onEsimeneVeerand = 0, onViimaneVeerand = 0, onPaike
= 0,
        onTeada = 0, onPooripaevTeada = 1, onTaiskuuTeada = 1, onKuuLoomineTeada = 1,
        onEsimeneVeerandTeada = 1, onViimaneVeerandTeada = 1;
    char strAsukoht[12];
    SDL_Rect abi;
    kuupaev k1, abi_k;
    k1 = k;

    SDL_FillRect(ekraan, NULL, SDL_MapRGB(ekraan->format, 255,255,255));


    for(i = 0; i < 40 && intLaius < ekraan->w; i++)
    {
        if(i > 0) k = saaKuupaev_lisaArv(k, 1.0);

        intPaev = intSaaNadalapaevKuupaevast(k);
        sprintf(strAsukoht, "img/n%d.bmp", intPaev);
        lisaPilt(ekraan, strAsukoht, intLaius, intSirviH);

        intPyha = onSirviPysivPyha(k);
        if(intPyha > 0 && intPyha != 25)
        {
```

```
            sprintf(strAsukoht, "img/p%d.bmp", intPyha);
            intAbi = intPyha - 1;
            lisaPilt2(ekraan, strAsukoht, intLaius - pPyhadArr[intAbi][0] + nPaevadArr[
intPaev][0], intSirviH - pPyhadArr[intAbi][2]);
        }else if(intPyha == 25 && !onJoulud)
        {
            sprintf(strAsukoht, "img/p%d.bmp", intPyha);
            intAbi = (k.paev - 21) * 17.5;
            intAbi = (int) intAbi;
            lisaPilt2(ekraan, strAsukoht,
                intLaius - intAbi,
                intSirviH - pPyhadArr[intPyha - 1][2]);
            //printf("intAbi: %d %d\n", intAbi, k.paev);
            onJoulud = 1;
        }

        if(intPyha == 25)
        {
            abi.x = intLaius + nPaevadArr[intPaev][0];
            abi.y = intSirviH - 7;
            abi.w = 3;
            abi.h = 7;
            SDL_FillRect(ekraan, &abi, SDL_MapRGB(ekraan->format,0,0,0));
        }


        intPyha = onSirviLiikuvPyha(k, &onTeada);
        if(!onTeada) onPooripaevTeada = 0;

        if(intPyha > 0 && intPyha != 8 || intPyha == 8 && !onKihlakud)
        {
            onPaike = 1;
            sprintf(strAsukoht, "img/L%d.bmp", intPyha);
            intAbi = intPyha - 1;
            lisaPilt2(ekraan, strAsukoht,
                intLaius - lPyhadArr[intAbi][0] + nPaevadArr[intPaev][0],
                intSirviH - lPyhadArr[intAbi][2]);

            if(intPyha == 8)
            {
                onKihlakud = 1;
            }
        }

        if(intPyha == 8)
        {
            abi.x = intLaius + nPaevadArr[intPaev][0];
            abi.y = intSirviH - 14;
            abi.w = 3;
            abi.h = 14;
            SDL_FillRect(ekraan, &abi, SDL_MapRGB(ekraan->format,0,0,0));
        }

        if(i == 0)
        {
            abi_k = saaKuupaev_lisaArv2(k, -3.0);
            abi_k = saaTaiskuu_kuupaev(k, &onTeada);
```

```
        if(!onTeada) onTaiskuuTeada = 0;
        arv_Taiskuu = intGregooriusVahe(abi_k, k) + 1;
        if(arv_Taiskuu < 2 || arv_Taiskuu > 3) arv_Taiskuu = 0;

        abi_k = saaKuupaev_lisaArv2(k, -3.0);
        abi_k = saaKuuLoomise_kuupaev(k, &onTeada);
        if(!onTeada) onKuuLoomineTeada = 0;
        arv_KuuLoomine = intGregooriusVahe(abi_k, k) + 1;
        if(arv_KuuLoomine < 2 || arv_KuuLoomine > 3) arv_KuuLoomine = 0;
    }

    //Täiskuu
    abi_k = saaTaiskuu_kuupaev(k, &onTeada);
    if(!onTeada) onTaiskuuTeada = 0;
    intAbi = intGregooriusVahe(k, abi_k);
    if(intAbi == 2)
    {
        onTaiskuu = 1;
        lisaPilt2(ekraan, "img/k4.bmp",
            intLaius - kFaasidArr[3][0] + nPaevadArr[intPaev][0],
            intSirviH + nPaevadArr[0][0]);
    }
    if(intAbi == 0)
    {
        lisaPilt2(ekraan, "img/k5.bmp",
            intLaius - kFaasidArr[4][0] + nPaevadArr[intPaev][0],
            intSirviH + nPaevadArr[0][0]);

        arv_Taiskuu = 1;
        onTaiskuu = 1;
    }
    if(arv_Taiskuu == 3)
    {
        lisaPilt2(ekraan, "img/k6.bmp",
            intLaius - kFaasidArr[5][0] + nPaevadArr[intPaev][0],
            intSirviH + nPaevadArr[0][0]);
        arv_Taiskuu = 0;
        onTaiskuu = 1;
    }
    if(arv_Taiskuu > 0) arv_Taiskuu++;

    //Kuu Loomine
    abi_k = saaKuuLoomise_kuupaev(k, &onTeada);
    if(!onTeada) onKuuLoomineTeada = 0;
    intAbi = intGregooriusVahe(k, abi_k);
    if(intAbi == 2)
    {
        onKuuLoomine = 1;
        lisaPilt2(ekraan, "img/k8.bmp",
            intLaius - kFaasidArr[7][0] + nPaevadArr[intPaev][0],
            intSirviH + nPaevadArr[0][0]);
    }
    if(intAbi == 0)
    {
        onKuuLoomine = 1;
        lisaPilt2(ekraan, "img/k1.bmp",
            intLaius - kFaasidArr[0][0] + nPaevadArr[intPaev][0],
```

```
                intSirviH + nPaevadArr[0][0]);

            arv_KuuLoomine = 1;
        }
        if(arv_KuuLoomine == 3)
        {
            onKuuLoomine = 1;
            lisaPilt2(ekraan, "img/k2.bmp",
                intLaius - kFaasidArr[1][0] + nPaevadArr[intPaev][0],
                intSirviH + nPaevadArr[0][0]);
            arv_KuuLoomine = 0;
        }
        if(arv_KuuLoomine > 0) arv_KuuLoomine++;

        //Esimene Veerand
        abi_k = saaKuuEsimeseVeerandi_kuupaev(k, &onTeada);
        if(!onTeada) onEsimeneVeerandTeada = 0;
        intAbi = intGregooriusVahe(k, abi_k);
        if(intAbi == 0)
        {
            onEsimeneVeerand = 1;
            lisaPilt2(ekraan, "img/k3.bmp",
                intLaius - kFaasidArr[2][0] + nPaevadArr[intPaev][0],
                intSirviH + nPaevadArr[0][0]);
        }

        //Viimane Veerand
        abi_k = saaKuuViimaseVeerandi_kuupaev(k, &onTeada);
        if(!onTeada) onViimaneVeerandTeada = 0;
        intAbi = intGregooriusVahe(k, abi_k);
        if(intAbi == 0)
        {
            onViimaneVeerand = 1;
            lisaPilt2(ekraan, "img/k7.bmp",
                intLaius - kFaasidArr[6][0] + nPaevadArr[intPaev][0],
                intSirviH + nPaevadArr[0][0]);
        }

        intLaius += nPaevadArr[intPaev][1] + 3;

    }

    abi.x = 0;
    abi.y = intSirviH - 2;
    abi.h = 2;
    abi.w = ekraan->w;
    SDL_FillRect(ekraan, &abi, SDL_MapRGB(ekraan->format, 0,0,0));
    abi.y = intSirviH + 61;
    SDL_FillRect(ekraan, &abi, SDL_MapRGB(ekraan->format, 0,0,0));

    lisaPilt(ekraan, "img/nooled.bmp", 240, 450); //235

    intAbi = 20;
    h = 120;
    if(onPaike)
    {
        lisaPilt_hw(ekraan, "img/paike.bmp", intAbi, h, &w, &intPyha);
```

```
        intAbi += w;
        if(!onPooripaevTeada)
            lisaPilt2(ekraan, "img/Hyyumark.bmp", intAbi - 30, h + intPyha - 30);
    }
    if(onKuuLoomine)
    {
        lisaPilt_hw(ekraan, "img/kuu_loomine.bmp", intAbi, h, &w, &intPyha);
        intAbi += w;
        if(!onKuuLoomineTeada)
            lisaPilt2(ekraan, "img/Hyyumark.bmp", intAbi - 27, h + intPyha - 30);
    }
    if(onEsimeneVeerand)
    {
        lisaPilt_hw(ekraan, "img/esimene_veerand.bmp", intAbi, h, &w, &intPyha);
        intAbi += w;
        if(!onEsimeneVeerandTeada)
            lisaPilt2(ekraan, "img/Hyyumark.bmp", intAbi - 27, h + intPyha - 30);
    }
    if(onTaiskuu)
    {
        lisaPilt_hw(ekraan, "img/taiskuu.bmp", intAbi, h, &w, &intPyha);
        intAbi += w;
        if(!onTaiskuuTeada)
            lisaPilt2(ekraan, "img/Hyyumark.bmp", intAbi - 27, h + intPyha - 30);
    }
    if(onViimaneVeerand)
    {
        lisaPilt(ekraan, "img/viimane_veerand.bmp", intAbi, h);
        intAbi += w;
        if(!onViimaneVeerandTeada)
            lisaPilt2(ekraan, "img/Hyyumark.bmp", intAbi - 27, h + intPyha - 30);
    }

    lisaKuupaevad(k1, k, ekraan, 10, ekraan->h - 20);
    k = gregoorius_sirviks(k);
    k1 = gregoorius_sirviks(k1);
    lisaKuupaevad(k1, k, ekraan, ekraan->w - 170, ekraan->h - 20);

    lisaPilt(ekraan, "img/pealkiri.bmp", 5, 0);
    lisaPilt(ekraan, "img/valju.bmp", ekraan->w - 90, 5);

    SDL_Flip(ekraan);
}
void lisaKuupaevad(kuupaev k1, kuupaev k2, SDL_Surface *img, int x, int y)
{
    char strPunkt[] = "img/nr_punkt.bmp";
    int w, h;

    x += lisaInt(k1.paev, 3,img, x, y);
    lisaPilt_hw(img, strPunkt, x, y, &w, &h);
    x += w;
    x += lisaInt(k1.kuu, 3,img, x, y);
    lisaPilt_hw(img, strPunkt, x, y, &w, &h);
    x += w;
    x += lisaInt(k1.aasta, 5,img, x, y);

    lisaPilt_hw(img, "img/nr_kriips.bmp", x + 3, y, &w, &h);
```

```
    x += w + 6;

    x += lisaInt(k2.paev, 3,img, x, y);
    lisaPilt_hw(img, strPunkt, x, y, &w, &h);
    x += w;
    x += lisaInt(k2.kuu, 3,img, x, y);
    lisaPilt_hw(img, strPunkt, x, y, &w, &h);
    x += w;
    x += lisaInt(k2.aasta, 5,img, x, y);
}
int lisaInt(int nr, int intPikkus, SDL_Surface *img, int x, int y)
{
    int ret = 0, i, w, h;
    char str[intPikkus], strAsukoht[12];

    sprintf(str, "%d", nr);

    for(i = 0; i < intPikkus; i++)
    {
        if(str[i] > 47 && str[i] < 58)
        {
            sprintf(strAsukoht, "img/nr%d.bmp", str[i] - 48);
            lisaPilt_hw(img, strAsukoht, x, y, &w, &h);
            ret += w;
            x += w;
        }

    }

    return(ret);
}
void lisaPilt2(SDL_Surface *ekraan, char strAsukoht[], int x, int y)
{
    SDL_Surface *img;
    img = SDL_LoadBMP(strAsukoht);

    Uint8 r, g, b;
    int _x, _y;

    if(SDL_MUSTLOCK(img)) { if(SDL_LockSurface(img) < 0) return; }

    for(_y = 0; _y < img->h; _y++)
    {
        for(_x = 0; _x < img->w; _x++)
        {
            SDL_GetRGB(getpixel(img, _x, _y), img->format, &r, &b, &g);
            if(r == g && g == b && b == 255)
            {
                putpixel(img, _x, _y, getpixel(ekraan, x + _x, y + _y));
            }

        }
    }

    if(SDL_MUSTLOCK(img)) SDL_UnlockSurface(img);

    SDL_Rect dest;
```

```
    dest.x = x;
    dest.y = y;
    SDL_BlitSurface(img, NULL, ekraan, &dest);
}
void lisaPilt(SDL_Surface *ekraan, char strAsukoht[], int x, int y)
{
    SDL_Surface *img;
    img = SDL_LoadBMP(strAsukoht);

    SDL_Rect dest;
    dest.x = x;
    dest.y = y;
    SDL_BlitSurface(img, NULL, ekraan, &dest);
}
void lisaPilt_hw(SDL_Surface *ekraan, char strAsukoht[], int x, int y, int *w, int *h)
{
    SDL_Surface *img;
    img = SDL_LoadBMP(strAsukoht);

    SDL_Rect dest;
    dest.x = x;
    dest.y = y;
    SDL_BlitSurface(img, NULL, ekraan, &dest);

    *w = img->w;
    *h = img->h;
}

Uint32 getpixel(SDL_Surface *surface, int x, int y)
{
    int bpp = surface->format->BytesPerPixel;
    /* Here p is the address to the pixel we want to retrieve */
    Uint8 *p = (Uint8 *)surface->pixels + y * surface->pitch + x * bpp;

    switch(bpp) {
    case 1:
        return *p;

    case 2:
        return *(Uint16 *)p;

    case 3:
        if(SDL_BYTEORDER == SDL_BIG_ENDIAN)
            return p[0] << 16 | p[1] << 8 | p[2];
        else
            return p[0] | p[1] << 8 | p[2] << 16;

    case 4:
        return *(Uint32 *)p;

    default:
        return 0;       /* shouldn't happen, but avoids warnings */
    }
}
void putpixel(SDL_Surface *surface, int x, int y, Uint32 pixel)
{
    int bpp = surface->format->BytesPerPixel;
```

```
    /* Here p is the address to the pixel we want to set */
    Uint8 *p = (Uint8 *)surface->pixels + y * surface->pitch + x * bpp;

    switch(bpp) {
    case 1:
        *p = pixel;
        break;

    case 2:
        *(Uint16 *)p = pixel;
        break;

    case 3:
        if(SDL_BYTEORDER == SDL_BIG_ENDIAN) {
            p[0] = (pixel >> 16) & 0xff;
            p[1] = (pixel >> 8) & 0xff;
            p[2] = pixel & 0xff;
        } else {
            p[0] = pixel & 0xff;
            p[1] = (pixel >> 8) & 0xff;
            p[2] = (pixel >> 16) & 0xff;
        }
        break;

    case 4:
        *(Uint32 *)p = pixel;
        break;
    }
}
```