Балтийский государственный технический университет «ВОЕНМЕХ»

им. Д. Ф. Устинова

Кафедра И5

«Информационные системы и программная инженерия»

Практическое задание № 1

по дисциплине «Программирование на ЯВУ»

на тему «Графический режим ввода-вывода в языках С и С++   
с использованием SDL»

Выполнил:

Студент Дубровский В.И.

Группа И582

Преподаватель:

Спирин Д.О.

Санкт-Петербург

2019

**Вариант 8**

*Цель работы* – познакомиться с возможностями создания программ с графическим пользовательским интерфейсом на языках С и C++ с использованием библиотек семейства SDL, научиться строить графики математических функций и создавать движущиеся изображения.

Задание 1. Построить график функции  при .

*Расчеты:* для построения точек графика использовалась формула *xэкр* = *x0экр* + floor(*x*×*MX*)

*Текст программы:*

myexc.h

#include <exception>

#include <string>

class Myexception : public std::exception

{

private:

std::string errName;

int errValue;

public:

Myexception(std::string mess, int i)

{

this->errName=mess;

this->errValue=i;

};

int GetValue(void) { return errValue; }

std::string GetErrM(void) { return errName; }

};

sdlmainclass.h

#include <SDL.h>

#include <SDL\_draw.h>

#include <SDL\_image.h>

#include <SDL\_ttf.h>

#include "myexc.h"

/////////////////

////////includes for Interval

////////////////

#include <string>

#include <sstream>

using namespace std;

/////////////////

////////includes for Grapher

////////////////

#include <math.h>

///////////////////////////

////////////////////

///////support func////////

///////////////////

//////////////////////////

double mathFunc(double x);

bool check(double &x1, double &x2);

std::string double\_string(double digit, int accuracy);

///////////////////////////

////////////////////

//////////////////////////

///////////////////

//////////////////////////

class SDLmain

{

protected:

char \*font;

TTF\_Font \*f[120];

SDL\_Surface \*screen;

const int WIDHT;

const int HEIGHT;

const int BPP;

//func for bliting on main surface

void applysurface(SDL\_Surface \*source, Sint16 x, Sint16 y);

public:

SDLmain(int w, int h, int b) : WIDHT(w), HEIGHT(h), BPP(b)

{

if (SDL\_Init(SDL\_INIT\_EVERYTHING) == -1) throw Myexception(SDL\_GetError(), -1);

if (TTF\_Init() == -1) throw Myexception(TTF\_GetError(), -1);

int imgflag= IMG\_INIT\_PNG|IMG\_INIT\_JPG;

if (IMG\_Init(imgflag) == -1) throw Myexception(IMG\_GetError(), -1);

screen = SDL\_SetVideoMode(WIDHT,HEIGHT,BPP, SDL\_HWSURFACE);

if(screen == NULL) throw Myexception(SDL\_GetError(), 101);

font\_init("FreeSerif.ttf");

for(int i=0;i<120; i++)

{

f[i]=NULL;

}

};

~SDLmain()

{

SDL\_FreeSurface(screen);

SDL\_Quit();

};

/\*func for bliting text on main surface

\*x and y - screen coordinates

\*message - c.type string

\*size - size of font

\*tcolor - color with type {R,G,B}\*/

void text\_surf(Sint16 x, Sint16 y, const char \*message, int size, SDL\_Color tcolor);

/\*func for bliting image on main surface

\*x and y - screen coordinates

\*dirimg- direction of ur image\*/

void image\_surf(Sint16 x, Sint16 y, const char \*dirimg);

/\*func for bliting image on main surface

with colorkeying

\*x and y - screen coordinates

\*dirimg- direction of ur image

\*fone - int array {R,G,B}\*/

void image\_surf(Sint16 x, Sint16 y, const char \*dirimg, int fone[]);

/\*simple event

press ESC or red cross\*/

void exit\_event(void);

//init font path

void font\_init(char \*path) { font=path; };

//blip main surface with typed color {R,G,B}

void blipscreen(int color[]);

//flip screen

void FLIP(void) { SDL\_Flip(screen); };

};

///////////////////////////

////////////////////

//////////////////////////

///////////////////

//////////////////////////

class Interval : public SDLmain

{

public:

Interval(int w, int h, int b) : SDLmain(w,h,b)

{};

/\* keyevent///bliting main surface

\*x and y - screen coordinates

\*size - size of font

\*tcolor - color with type {R,G,B}

!dont forgot to init font\*/

bool keyevent(Sint16 x, Sint16 y, int size, SDL\_Color color, double &out\_x);

/\* init\_x

string - double in format x

double - out x

return flag \*/

bool init\_x(string, double &);

//Interface

void interface(void);

};

///////////////////////////

////////////////////

//////////////////////////

///////////////////

//////////////////////////

class Grapher : public SDLmain

{

private:

struct coordinates

{

double x1;

double x2;

double ymax;

double ymin;

} coor;

struct zoomfactor

{

double xzoom;

double yzoom;

double DX;

double cx;

double cy;

} zoom;

public:

Grapher(int w, int h, int b, double x1, double x2, double (\*mathF)(double)) : SDLmain(w, h, b)

{

coor.x1=x1;

coor.x2=x2;

coor.ymin=mathFunc(coor.x1);

coor.ymax=mathFunc(coor.x2);

zoom.DX=fabs(x2-x1)/(WIDHT\*10);

for(double x=coor.x1; x<=coor.x2; x+=zoom.DX)

{

if(coor.ymin>mathFunc(x) && coor.ymin>=-50) coor.ymin=mathFunc(x);

if(coor.ymax<mathFunc(x) && coor.ymax<=50) coor.ymax=mathFunc(x);

if(coor.ymin<-50) coor.ymin=-50;

if(coor.ymax>50) coor.ymax=50;

}

zoom.xzoom=WIDHT/fabs(x2-x1);

zoom.yzoom=HEIGHT/(coor.ymax-coor.ymin);

zoom.cx = x1;

zoom.cy = coor.ymin;

};

void makegraphic(void);

/\*need to remake

!!!!!!!!!!!!!!!!!!!!!!!!\*/

void axis(void);

bool arrow\_event(void);

friend double mathFunc(double);

bool circle(void);

};

main.cpp

#include <iostream>

#include "sdlmainclass.h"

using namespace std;

int main() {

try{

SDLmain \*p;

SDL\_Event event;

double x1, x2;

bool flag1=true, flag2=true, flag3=true;

bool flag=true;

system("ls");

Again:

while(flag1 || flag2 || flag3)

{

p = new Interval(600,400,32);

((Interval\*)p)->interface();

flag1=((Interval\*)p)->keyevent(70,120,32, {0,0,0}, x1);

flag2=((Interval\*)p)->keyevent(70,200,32, {0,0,0}, x2);

flag3=check(x1,x2);

delete(p);

}

flag1=flag2=flag3=true;

p = new Grapher(1600, 900, 32, x1, x2, mathFunc);

((Grapher\*)p)->makegraphic();

flag=((Grapher\*)p)->arrow\_event();

if (flag) goto Again;

return 0;

}

catch(Myexception &e){

cerr << "Error: " << e.GetErrM() << endl;

return e.GetValue();

};

}

sdlmainclass.cpp

#include "sdlmainclass.h"

///////////////////////////

////////////////////

///////support func////////

///////////////////

//////////////////////////

double mathFunc(double x)

{

return x\*x-6/(x-1)+3;

}

bool check(double &x1, double &x2)

{

if(x1==x2) return true;

if(x1>x2) {

double temp=x2;

x2=x1;

x1=temp;

}

if(fabs(x2-x1)<0.25 || fabs(x2-x1)>=50) return true;

return false;

}

std::string double\_string(double digit, int accuracy)

{

string str=to\_string(digit);

str.resize(str.find('.')+accuracy+1);

return str;

}

///////////////////////////

////////////////////

//////////////////////////

///////////////////

//////////////////////////

void SDLmain::text\_surf(Sint16 x, Sint16 y, const char \*message, int size, SDL\_Color tcolor)

{

if(f[size]==NULL){

f[size]=TTF\_OpenFont(font, size);

if (f[size] == NULL) throw Myexception(TTF\_GetError(), 101);

}

SDL\_Surface \*text=TTF\_RenderUTF8\_Solid(f[size], message, tcolor);

if (text == NULL) throw Myexception(TTF\_GetError(), 101);

applysurface(text, x, y);

SDL\_FreeSurface(text); //need it???

}

void SDLmain::applysurface(SDL\_Surface \*source, Sint16 x, Sint16 y)

{

SDL\_Rect offset;

offset.x=x;

offset.y=y;

if(SDL\_BlitSurface(source, NULL, screen, &offset) == -1) throw Myexception(SDL\_GetError(), -1);

}

void SDLmain::image\_surf(Sint16 x, Sint16 y, const char \*dirimg)

{

SDL\_Surface \*temp=IMG\_Load(dirimg);

if (temp == NULL) throw Myexception(IMG\_GetError(), 101);

SDL\_Surface \*formatimg=SDL\_ConvertSurface(temp, screen->format, NULL);

if (formatimg == NULL) throw Myexception(IMG\_GetError(), 101);

SDL\_FreeSurface(temp);

applysurface(formatimg, x, y);

SDL\_FreeSurface(formatimg);// aakka

}

void SDLmain::image\_surf(Sint16 x, Sint16 y, const char \*dirimg, int fone[])

{

SDL\_Surface \*temp=IMG\_Load(dirimg);

if (temp == NULL) throw Myexception(IMG\_GetError(), 101);

SDL\_Surface \*formatimg=SDL\_ConvertSurface(temp, screen->format, NULL);

if (formatimg == NULL) throw Myexception(IMG\_GetError(), 101);

Uint32 colorkey = SDL\_MapRGB(formatimg->format, fone[0], fone[1], fone[2]);

SDL\_SetColorKey(formatimg, SDL\_SRCCOLORKEY, colorkey);

SDL\_FreeSurface(temp);

applysurface(formatimg, x, y);

SDL\_FreeSurface(formatimg);// aakka

}

void SDLmain::exit\_event(void)

{

SDL\_Event event;

bool quit=true;

while (quit)

{

while(SDL\_PollEvent(&event))

{

if(event.key.keysym.sym == SDLK\_ESCAPE ||

event.type == SDL\_QUIT) quit=false;

}

}

SDL\_PumpEvents();

}

void SDLmain::blipscreen(int color[])

{

SDL\_FillRect(screen, NULL, SDL\_MapRGB(screen->format, color[0], color[1], color[2]));

}

///////////////////////////

////////////////////

//////////////////////////

///////////////////

//////////////////////////

bool Interval::keyevent(Sint16 x, Sint16 y, int size, SDL\_Color color, double &out\_x)

{

SDL\_Event event;

bool flag=true;

string str;

image\_surf(WIDHT-x\*3, y-20, "arrow.png");

SDL\_Flip(screen);

while(flag)

{

while(SDL\_PollEvent(&event))

{

if(event.key.state == 0) continue;

if(event.key.keysym.sym == 13) flag = false; // enter

if(event.key.keysym.sym == SDLK\_BACKSPACE)

{

if (str.size()==0) continue;

str.pop\_back();

Draw\_FillRect(screen,x+str.size()\*size/2, y, size/2, size\*1.25, 0xFFFFFF);

SDL\_Flip(screen);

}

if(event.key.keysym.sym >= 31 && event.key.keysym.sym <= 118)

{

if(str.size()!=9)

{

str += event.key.keysym.sym;

text\_surf(x , y, str.c\_str(), size, color);

SDL\_Flip(screen);

}

}

}

}

SDL\_PumpEvents();

Draw\_FillRect(screen, WIDHT-x\*3, y-20, 120, 70, 0xFFFFFF);

return init\_x(str,out\_x);

}

void Interval::interface(void)

{

blipscreen(new int[3] {255,255,255});

image\_surf(0, 0, "func.png");

text\_surf(270,50, "Введите интервал:", 40, {0,0,0});

text\_surf(10, 120, "x1=", 32, {0,0,0});

text\_surf(10, 200, "x2=", 32, {0,0,0});

SDL\_Flip(screen);

}

bool Interval::init\_x(string s, double &x)

{

istringstream ss(s);

if(ss >> x) return false;

return true;

}

///////////////////////////

////////////////////

//////////////////////////

///////////////////

//////////////////////////

void Grapher::makegraphic(void)

{

double y;

Sint16 xscreen, yscreen;

for(double x=coor.x1; x<=coor.x2; x+=zoom.DX)

{

y=mathFunc(x);

if(y<coor.ymin|| y>coor.ymax) continue;

xscreen=floor((x-zoom.cx)\*zoom.xzoom);

yscreen=HEIGHT-floor((y-zoom.cy)\*zoom.yzoom);

if(yscreen>=HEIGHT) continue;

Draw\_Pixel(screen, xscreen, yscreen, 0xFF00FF);

}

axis();

SDL\_Flip(screen);

}

void Grapher::axis(void)

{

Sint16 x1=0,x2=0,y1=0,y2=HEIGHT-1;

double dY=fabs(coor.ymax-coor.ymin)/11;

double ynums=coor.ymin;

if(coor.x1>0)

{

Draw\_Line(screen,x1,0,20,30,0xFFFFFF);

for(int i=HEIGHT-HEIGHT/10-1; i>20; i-=HEIGHT/10)

{

ynums+=dY;

text\_surf(WIDHT/100,i+HEIGHT/100, double\_string(ynums, 2).c\_str(), 20, {255,255,255});

Draw\_Line(screen,x1, i, 10, i, 0xFFFFFF);

}

text\_surf(30, 20, "Y", 30, {255,255,255});

}

else if(coor.x2<0)

{

x1=WIDHT-1;

Draw\_Line(screen, x1,0, x1-20, 30, 0xFFFFFF);

for(int i=HEIGHT-HEIGHT/10-1; i>20; i-=HEIGHT/10)

{

ynums+=dY;

text\_surf(WIDHT-WIDHT/25,i+HEIGHT/100, double\_string(ynums, 2).c\_str(), 20, {255,255,255});

Draw\_Line(screen,x1, i, x1-20, i, 0xFFFFFF);

}

text\_surf(x1-WIDHT/25, 20, "Y", 30, {255,255,255});

}

else

{

x1=floor(-1\*zoom.cx\*zoom.xzoom);

Draw\_Line(screen, x1, 0, x1-20, 30, 0xFFFFFF);

Draw\_Line(screen, x1, 0, x1+20, 30, 0xFFFFFF);

for(int i=HEIGHT-HEIGHT/10-1; i>20; i-=HEIGHT/10)

{

ynums+=dY;

text\_surf(x1+WIDHT/100,i+HEIGHT/100, double\_string(ynums, 2).c\_str(), 20, {255,255,255});

Draw\_Line(screen,x1-20, i, x1+20, i, 0xFFFFFF);

}

text\_surf(x1+WIDHT/25, 20, "Y", 30, {255,255,255});

}

Draw\_Line(screen,x1,y1,x1,y2,0xFFFFFF);

x1=0; x2=WIDHT-1;

dY=fabs(coor.x2-coor.x1)/21;

ynums=coor.x1;

if(coor.ymin>0) {

y1=HEIGHT-1;

Draw\_Line(screen, x2-30, y1-20, x2,y1, 0xFFFFFF);

for(int i=WIDHT/20; i<WIDHT-20; i+=WIDHT/20)

{

ynums+=dY;

text\_surf(i+10, y1-50, double\_string(ynums, 2).c\_str(), 20, {255,255,255});

Draw\_Line(screen,i, y1, i, y1-20, 0xFFFFFF);

}

text\_surf(x2-30, y1-60, "X", 30, {255,255,255});

}

else if(0>coor.ymax) {

y1=0;

Draw\_Line(screen, x2-30, 20, x2, y1, 0xFFFFFF);

for(int i=WIDHT/20; i<WIDHT-20; i+=WIDHT/20)

{

ynums+=dY;

text\_surf(i+10, y1+20, double\_string(ynums, 2).c\_str(), 20, {255,255,255});

Draw\_Line(screen,i, y1, i, y1+20, 0xFFFFFF);

}

text\_surf(x2-30, y1+30, "X", 30, {255,255,255});

}

else

{

y1=HEIGHT-floor(-1\*zoom.cy\*zoom.yzoom);

if(y1+20>HEIGHT) Draw\_Line(screen, x2-30, HEIGHT-(HEIGHT-y1), x2, y1, 0xFFFFFF);

else Draw\_Line(screen, x2-30, y1+20, x2, y1, 0xFFFFFF);

if(y1-20<0) Draw\_Line(screen, x2-30, 20-y1, x2, y1, 0xFFFFFF);

else Draw\_Line(screen, x2-30, y1-20, x2, y1, 0xFFFFFF);

for(int i=WIDHT/20; i<WIDHT-20; i+=WIDHT/20)

{

ynums+=dY;

if(y1<20){

text\_surf(i+10, y1+20, double\_string(ynums, 2).c\_str(), 20, {255,255,255});

Draw\_Line(screen,i, 20-y1, i, y1+20, 0xFFFFFF);

}

else if(y1>HEIGHT-20)

{

text\_surf(i+10, y1-20, double\_string(ynums, 2).c\_str(), 20, {255,255,255});

Draw\_Line(screen,i, y1-HEIGHT-20, i, y1+20, 0xFFFFFF);

}

else{

text\_surf(i+10, y1+20, double\_string(ynums, 2).c\_str(), 20, {255,255,255});

Draw\_Line(screen,i, y1-20, i, y1+20, 0xFFFFFF);

}

}

text\_surf(x2-30, y1+30, "X", 30, {255,255,255});

}

Draw\_Line(screen,x1,y1,x2,y1,0xFFFFFF);

}

bool Grapher::arrow\_event(void)

{

SDL\_Event event;

bool flag=true;

double dy=fabs(coor.ymax-coor.ymin)/3;

double dx=fabs(coor.x2-coor.x1)/3;

while(flag)

{

while(SDL\_PollEvent(&event))

{

if(event.key.state == 0) continue;

if(event.type == SDL\_QUIT) return false;

switch(event.key.keysym.sym)

{

case SDLK\_UP:

coor.ymin+=dy;

coor.ymax+=dy;

zoom.cy=coor.ymin;

break;

case SDLK\_DOWN:

coor.ymin-=dy;

coor.ymax-=dy;

break;

case SDLK\_RIGHT:

coor.x1+=dx;

coor.x2+=dx;

break;

case SDLK\_LEFT:

coor.x1-=dx;

coor.x2-=dx;

break;

case 49:

if(fabs(coor.x2-coor.x1)<0.5) break;

coor.x2\*=0.75;

coor.ymax\*=0.75;

coor.x1\*=0.75;

coor.ymin\*=0.75;

zoom.xzoom=WIDHT/fabs(coor.x2-coor.x1);

zoom.yzoom=HEIGHT/fabs(coor.ymax-coor.ymin);

break;

case 50:

if(fabs(coor.ymax-coor.ymin)>200) break;

coor.x2\*=1.25;

coor.ymax\*=1.25;

coor.x1\*=1.25;

coor.ymin\*=1.25;

zoom.xzoom=WIDHT/fabs(coor.x2-coor.x1);

zoom.yzoom=HEIGHT/fabs(coor.ymax-coor.ymin);

break;

case 13:

return true;

case SDLK\_ESCAPE:

return false;

}

zoom.cy=coor.ymin;

zoom.cx=coor.x1;

dy=fabs(coor.ymax-coor.ymin)/3;

dx=fabs(coor.x2-coor.x1)/3;

blipscreen(new int[3] {0,0,0});

makegraphic();

}

}

SDL\_PumpEvents();

}

bool Grapher::circle(void)

{

SDL\_Event event;

SDL\_PumpEvents();

while(SDL\_PollEvent(&event))

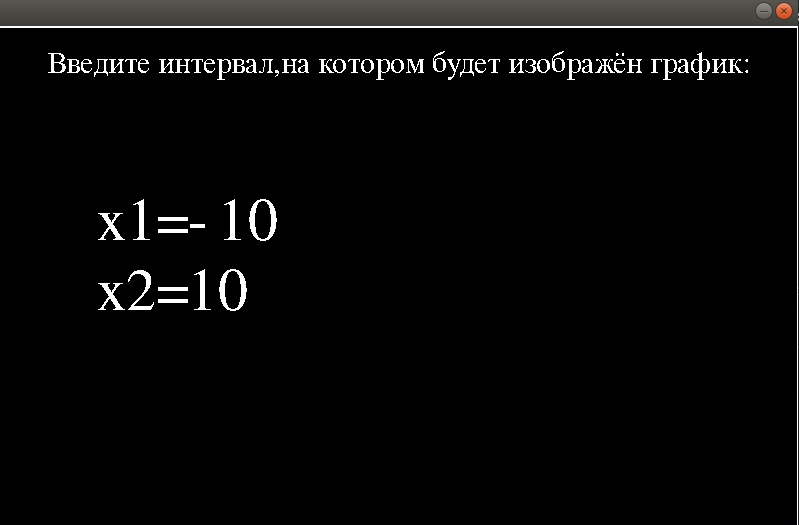
{

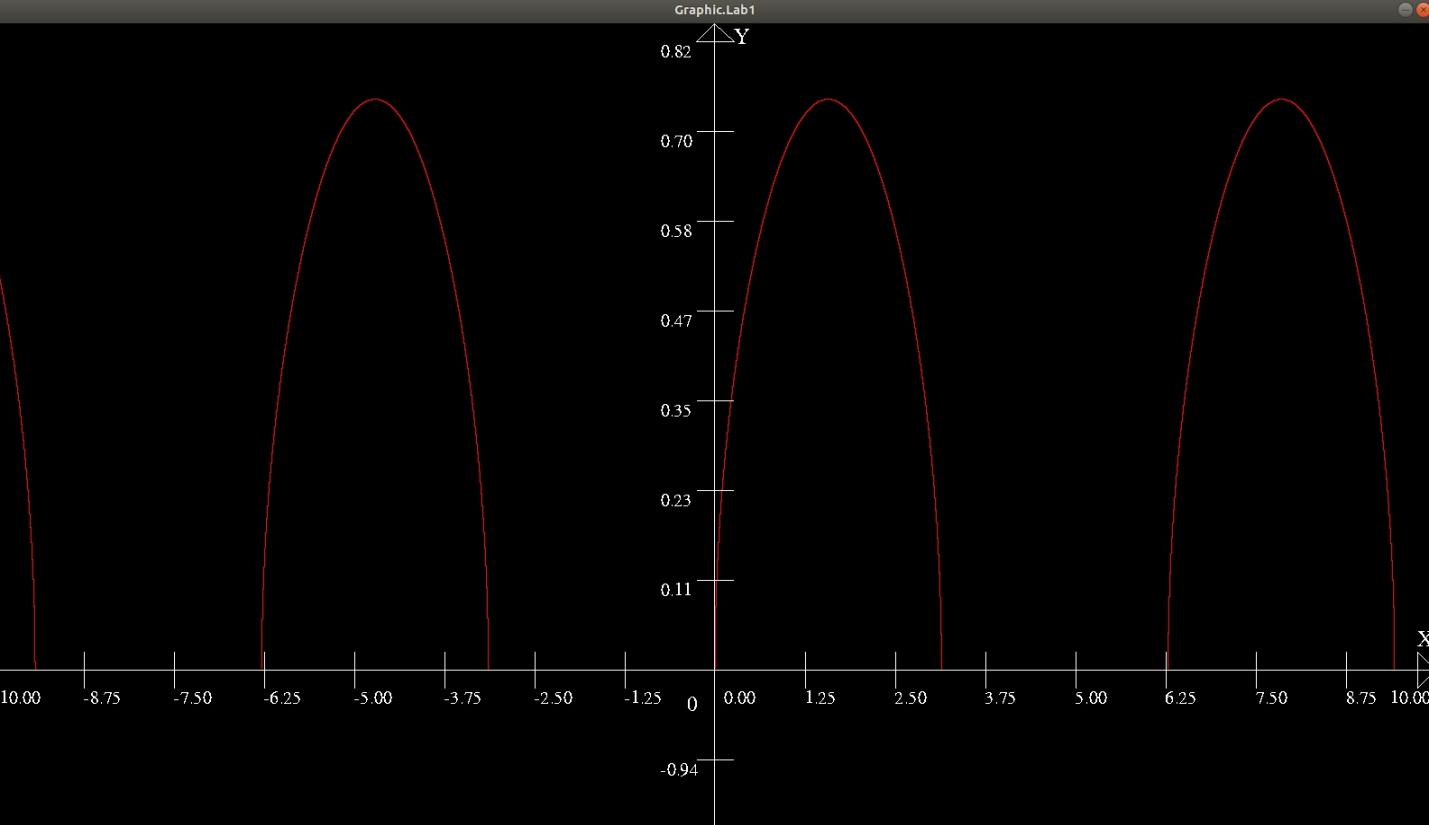
if(event.key.keysym.sym == 13) return true;

if(event.type == SDL\_QUIT) return false;

}

}*Результат работы:*

**



Задание 2*.* Изобразить на экране полет самолета на заданной высоте и посадку его со снижением до касания земли и замедлением до полной остановки.

*Расчеты :* для изображения дуги посадки самолёта использовалось квадратичное уравнение тела, брошенного горизонтально y=y0-g\*t^2/2. Траектория – парабола.

*Текст программы:*

main.cpp

#include "myclass.h"

#include <iostream>

#include <math.h>

class FirstWin : public SDLmain

{

public:

FirstWin(int w, int h, int b) : SDLmain(w,h,b)

{};

double keyboard\_event(Sint16 xbegin, Sint16 ybegin, Sint16 sizetext, int line); //shitty func NEED tO do better

};

int main() {

try{

double height;

FirstWin obj(800,400,32);

obj.text\_surf(10, 10, "Введите начальную высоту полёта самолёта в метрах:", 26, {255,255,255});

height=obj.keyboard\_event(40,40,26, 11);

Plane airplane(1600,800,32, height);

if (height<=0 || height>37000) {

airplane.text\_surf(50,300,"Высота полёта введена некорректно!", 100, {255,0,0});

obj.exit\_event();

exit(1);

}

airplane.image\_surf(0,airplane.HEIGHT/2,"/home/vladislav/Загрузки/air1.png");

airplane.gif\_surf();

airplane.image\_surf(0, 0,"/home/vladislav/vscoderep/slab2/back.jpg");

airplane.falling();

airplane.exit\_event();

return 0;

}

catch(Myexception &e)

{

cerr << "Error: " << e.GetErrM() << endl;

return e.GetValue();

}

}

double FirstWin::keyboard\_event(Sint16 xbegin, Sint16 ybegin, Sint16 sizetext, int line)

{

SDL\_Event event;

bool quit=true;

bool flag=false;

bool flag1=false;

char count=-1;

char \*num=(char\*)malloc(line);

memset(num,'\0', line);

SDL\_Flip(screen);

while(quit)

{

while(SDL\_PollEvent(&event))

{

if(event.key.state == 0) continue;

if(event.key.keysym.sym == 13 && strlen(num)!=0) quit = false ;

if((event.key.keysym.sym>=48 && event.key.keysym.sym<=57) || event.key.keysym.sym=='.')

{

if (count==line-2);

else{

if(count!=-1 && event.key.keysym.sym=='-');

else if (flag1 && event.key.keysym.sym=='-');

else if (flag && event.key.keysym.sym=='.');

else {

if(event.key.keysym.sym=='-') flag1=true;

if (event.key.keysym.sym=='.') flag=true;

char \*s;

count++;

s=(char\*)calloc(sizeof(char), 2);

num[count]=s[0]=event.key.keysym.sym;

s[1]='\0';

text\_surf(xbegin, ybegin, s, sizetext, {255,255,255});

free(s);

xbegin+=sizetext/2;

SDL\_Flip(screen);

}

}

}

if(event.key.keysym.sym==SDLK\_BACKSPACE)

{

if (count==-1);

else{

if(num[count]=='.') flag=false;

if(num[count]=='-') flag1=false;

xbegin-=sizetext/2;

Draw\_FillRect(screen, xbegin ,ybegin, sizetext/2, sizetext, 0x0000000);

num[count]='\0';

count--;

SDL\_Flip(screen);

}

}

}

}

SDL\_PumpEvents();

double point=0;

char flag3;

int dotind=count+1;

int i=0;

if(num[0]=='-') {

i++;

flag3=-1 ;

}

else flag3=1;

{

char \*flag2;

if((flag2=strchr(num,'.'))!=NULL) dotind=flag2-num; //get index of dot

}

for(;i<=count; i++){

if(i<dotind) point+=(num[i]-48)\*pow(10,dotind-i-1);

else if(i>dotind) point+=(num[i]-48)\*pow(10, dotind-i);

}

return point\*flag3;

}

myclass.cpp

#include "myclass.h"

void SDLmain::text\_surf(Sint16 x, Sint16 y, const char \*message, int size, SDL\_Color tcolor)

{

TTF\_Font \*ptr=TTF\_OpenFont(font, size);

if (ptr == NULL) throw Myexception(TTF\_GetError(), 101);

SDL\_Surface \*text=TTF\_RenderUTF8\_Solid(ptr, message, tcolor);

if (text == NULL) throw Myexception(TTF\_GetError(), 101);

applysurface(text, x, y);

SDL\_FreeSurface(text); //need it???

//SDL\_Flip(screen);

}

void SDLmain::applysurface(SDL\_Surface \*source, Sint16 x, Sint16 y)

{

SDL\_Rect offset;

offset.x=x;

offset.y=y;

if(SDL\_BlitSurface(source, NULL, screen, &offset) == -1) throw Myexception(SDL\_GetError(), -1);

}

void SDLmain::image\_surf(Sint16 x, Sint16 y, const char \*dirimg)

{

SDL\_Surface \*temp=IMG\_Load(dirimg);

if (temp == NULL) throw Myexception(IMG\_GetError(), 101);

SDL\_Surface \*formatimg=SDL\_ConvertSurface(temp, screen->format, NULL);

if (formatimg == NULL) throw Myexception(IMG\_GetError(), 101);

SDL\_FreeSurface(temp);

applysurface(formatimg, x, y);

SDL\_FreeSurface(formatimg);// aakka

}

void SDLmain::image\_surf(Sint16 x, Sint16 y, const char \*dirimg, int fone[])

{

SDL\_Surface \*temp=IMG\_Load(dirimg);

if (temp == NULL) throw Myexception(IMG\_GetError(), 101);

SDL\_Surface \*formatimg=SDL\_ConvertSurface(temp, screen->format, NULL);

if (formatimg == NULL) throw Myexception(IMG\_GetError(), 101);

Uint32 colorkey = SDL\_MapRGB(formatimg->format, fone[0], fone[1], fone[2]);

SDL\_SetColorKey(formatimg, SDL\_SRCCOLORKEY, colorkey);

SDL\_FreeSurface(temp);

applysurface(formatimg, x, y);

SDL\_FreeSurface(formatimg);// aakka

}

void SDLmain::exit\_event(void)

{

SDL\_Event event;

bool quit=true;

while (quit)

{

while(SDL\_PollEvent(&event))

{

if(event.key.keysym.sym == SDLK\_ESCAPE ||

event.type == SDL\_QUIT) quit=false;

}

}

SDL\_PumpEvents();

}

void SDLmain::blipscreen(int color[])

{

SDL\_FillRect(screen, NULL, SDL\_MapRGB(screen->format, color[0], color[1], color[2]));

}

///////////////////////////

////////////////////

//////////////////////////

///////////////////

//////////////////////////

void Plane::falling(void)

{

double x, y;

Sint16 screenX, screenY, scrXbegin=0, scrYbegin=HEIGHT-120;

double xmin=0, xmax=V0\*sqrt(2\*h/10), ymin=0, ymax=h;

double Xscale=WIDHT/(2\*fabs(xmax-xmin));

double Yscale=HEIGHT\*0.75/fabs(ymax-ymin);

double DX=pow(Xscale, -1)\*2;

string curH;

for(double x=xmin; x<=xmax; x+=DX)

{

y=mathfunc(x);

screenX=floor(x\*Xscale + scrXbegin);

screenY=floor(scrYbegin-y\*Yscale);

image\_surf(0,0,"/home/vladislav/vscoderep/slab2/back.jpg");

image\_surf(screenX-75,screenY-75,"/home/vladislav/Загрузки/air3.png",new int[3] {0,0,0});

curH = "H = " + double\_string(y,1) + " м.";

text\_surf(WIDHT\*0.85,HEIGHT\*0.33, curH.c\_str(), 30, {0,0,0});

time(y);

SDL\_Delay(floor(ymax/1000));

SDL\_Flip(screen);

}

string ts="T = " + double\_string(curT, 2) + " c.";

for(;screenX<WIDHT-75;screenX+=12)

{

image\_surf(0,0,"/home/vladislav/vscoderep/slab2/back.jpg");

image\_surf(screenX-75,screenY-75,"/home/vladislav/Загрузки/air3.png",new int[3] {0,0,0});

text\_surf(WIDHT\*0.85,HEIGHT\*0.25, ts.c\_str(), 30, {0,0,0});

text\_surf(WIDHT\*0.85,HEIGHT\*0.33, "H = 0.0 м.", 30, {0,0,0});

SDL\_Flip(screen);

}

SDL\_Flip(screen);

}

double Plane::mathfunc(double x)

{

return h-10\*x\*x/(2\*V0\*V0);

}

void Plane::time(double y)

{

curT=twhole-sqrt(2\*y/10)\*V0;

string ts=double\_string(curT, 2);

ts= "T = " + ts + " c.";

text\_surf(WIDHT\*0.85,HEIGHT\*0.25, ts.c\_str(), 30, {0,0,0});

}

std::string double\_string(double digit, int accuracy)

{

string str=to\_string(digit);

str.resize(str.find('.')+accuracy+1);

return str;

}

void Plane::gif\_surf(void)

{

SDL\_Event event;

string str="/home/vladislav/vscoderep/slab2/back/fe2b3d46f0974263a77962b8304ca8f3-";

int flag=71;

bool quit=true;

while(quit)

{

if(flag==-1) flag=71;

str.append(double\_string(flag,0));

str.append("jpeg");

image\_surf(0,0,str.c\_str());

image\_surf(0, HEIGHT\*0.35,"/home/vladislav/Загрузки/air3.png", new int[3] {0,0,0});

str="/home/vladislav/vscoderep/slab2/back/fe2b3d46f0974263a77962b8304ca8f3-";

flag--;

SDL\_Delay(40);

SDL\_Flip(screen);

while(SDL\_PollEvent(&event))

{

if(event.key.state == 0) continue;

if(event.key.keysym.sym==13 ) quit=false;

}

}

}

Myexc.cpp

#include "Myexc.h"

Myexception::Myexception(string errName, int errValue)

{

this->errName = errName;

this->errValue = errValue;

}

int Myexception::GetValue(void) { return errValue; }

string Myexception::GetErrM(void) { return errName; }

Myexc.h

using namespace std;

#include <exception>

#include <string>

class Myexception :public exception

{

private:

string errName;

int errValue;

public:

Myexception(string errName, int errValue);

int GetValue(void);

string GetErrM(void);

};

myclass.h

#include <SDL.h>

#include <SDL\_draw.h>

#include <SDL\_image.h>

#include <SDL\_ttf.h>

#include <math.h>

#include "Myexc.h"

#include <string>

using namespace std;

class SDLmain

{

protected:

SDL\_Surface \*screen;

char \*font;

void applysurface(SDL\_Surface \*source, Sint16 x, Sint16 y);

public:

const int WIDHT;

const int HEIGHT;

const int BPP;

SDLmain(int w, int h, int b) : WIDHT(w), HEIGHT(h), BPP(b)

{

if (SDL\_Init(SDL\_INIT\_EVERYTHING) == -1) throw Myexception(SDL\_GetError(), -1);

if (TTF\_Init() == -1) throw Myexception(TTF\_GetError(), -1);

int imgflag= IMG\_INIT\_PNG|IMG\_INIT\_JPG;

if (IMG\_Init(imgflag) == -1) throw Myexception(IMG\_GetError(), -1);

screen = SDL\_SetVideoMode(WIDHT,HEIGHT,BPP, SDL\_HWSURFACE);

if(screen == NULL) throw Myexception(SDL\_GetError(), 101);

font\_init("/usr/share/fonts/truetype/freefont/FreeSerif.ttf");

};

~SDLmain()

{

SDL\_FreeSurface(screen);

SDL\_Quit();

};

void text\_surf(Sint16 x, Sint16 y, const char \*message, int size, SDL\_Color tcolor);

void image\_surf(Sint16 x, Sint16 y, const char \*dirimg);

void image\_surf(Sint16 x, Sint16 y, const char \*dirimg, int fone[]);

void exit\_event(void);

void font\_init(char \*path) { font=path; };

void blipscreen(int color[]);

};

///////////////////////////

////////////////////

//////////////////////////

///////////////////

//////////////////////////

class Plane :public SDLmain

{

private:

double h;

double V0;

double twhole;

double curT;

void time(double y);

double mathfunc(double x);

public:

Plane(int w, int h, int b, double y) : SDLmain(w, h, b)

{

this->h=y;

this->V0=200 ;

this->twhole=sqrt(2\*y/10)\*V0;

};

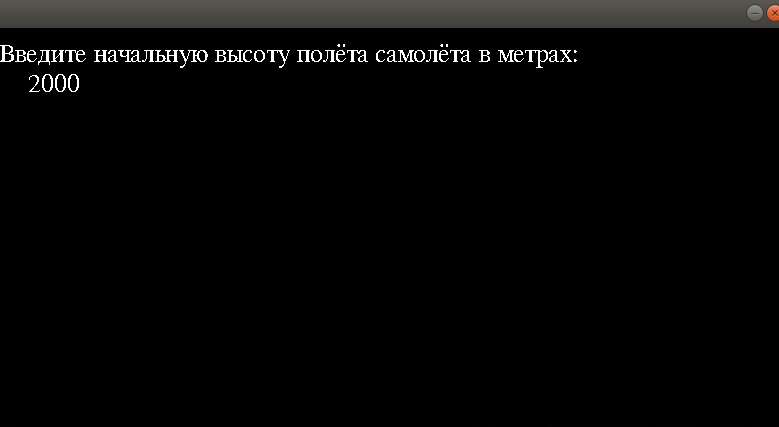
void falling(void);

void gif\_surf(void);

};

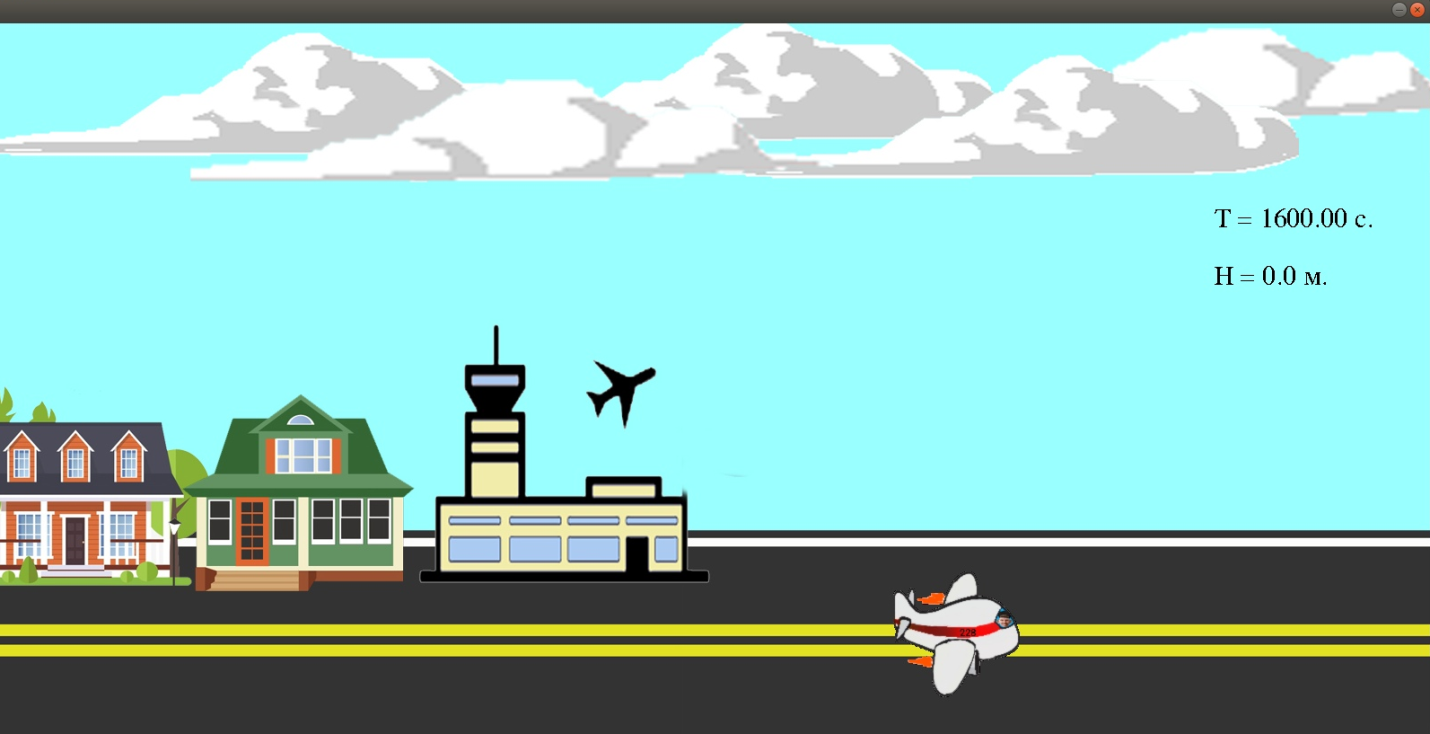
std::string double\_string(double digit, int accuracy);

*Результат работы:*

**

**

**

**