Белорусский государственный университет

Факультет прикладной математики и информатики

Кафедра технологий программирования

Жук Валерий Викторович

**Отчет**

**по учебной практике**

студента 1 курса 6 группы

**Преподаватель**

***Сатолина Анна Викторовна***

Ассистент кафедры технологий программирования ФПМИ

Минск, 2016

Оглавление

[Использование команд сопроцессора языка Ассемблер. 3](#_Toc453234316)

[Код: 4](#_Toc453234317)

[Скриншоты: 5](#_Toc453234318)

[Использование цепочечных команд языка Ассемблер. 5](#_Toc453234319)

[Код: 5](#_Toc453234320)

[Скриншоты: 8](#_Toc453234321)

[Использование внешних подпрограмм языка Ассемблер. 10](#_Toc453234322)

[Код: 10](#_Toc453234323)

[Скриншоты: 11](#_Toc453234324)

[WINAPI 11](#_Toc453234325)

[Скриншоты: 14](#_Toc453234326)

[Работа с графикой WINAPI 14](#_Toc453234327)

[Код: 14](#_Toc453234328)

[Скриншоты: 18](#_Toc453234329)

[Создание приложения WINAPI 19](#_Toc453234330)

[Код: 20](#_Toc453234331)

[Скриншоты: 28](#_Toc453234332)

[Создание приложения Windows Forms 31](#_Toc453234333)

[Код: 31](#_Toc453234334)

[Скриншоты: 42](#_Toc453234335)

[Создание приложения MFC 44](#_Toc453234336)

[Код: 44](#_Toc453234337)

[Скриншоты: 65](#_Toc453234338)

[Код: 66](#_Toc453234339)

[Скриншоты: 67](#_Toc453234340)

[Использование STL. Обработка контейнеров. Использование класса Vector. 68](#_Toc453234341)

[Код: 68](#_Toc453234342)

[Скриншоты: 70](#_Toc453234343)

[Адаптеры контейнеров 70](#_Toc453234344)

[Код: 70](#_Toc453234345)

[Скриншоты: 73](#_Toc453234346)

# Использование команд сопроцессора языка Ассемблер.

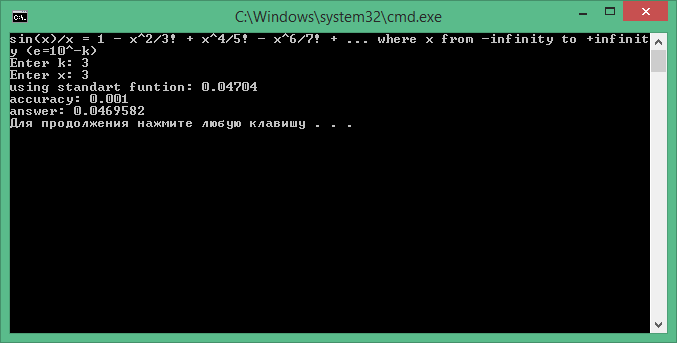
Вычислить ряд Тейлора используя команды сопроцессора.

sinx/x = 1 - x^2/3! + x^4/5! - x^6/7! + ... where x from -infinity to +infinity (e=10^-k)

## Код:

|  |  |  |
| --- | --- | --- |
| #include <iostream>  using namespace std;  double sinxDivx(const double x)  {  double func;  \_asm  {  finit  fld x  fsin  fdiv x  fstp func  }  return func;  }  double accuracy(const int k)  {  double acc = 1, tmp = 10;  \_asm  {  finit  mov ecx, k  fld acc  \_L1:  fdiv tmp  loop \_L1  fstp acc  }  return acc;  } | double teylor(const double x, const double acc)  {  double fuct = 1, num = 1, tmp, sum = 0, counter = 1, one = 1;  \_asm  {  finit  \_While:  fld num  fdiv fuct  fstp tmp  fld sum  fadd tmp  fstp sum  fld counter  fadd one  fstp counter  fld fuct  fmul counter  fstp fuct  fld counter  fadd one  fstp counter  fld fuct  fmul counter  fstp fuct  fld num  fmul x  fmul x  fchs  fstp num  fld num  fdiv fuct  fabs  fcom acc  fstsw ax  sahf  fstp tmp  jc \_Exit  jmp \_While  \_Exit:  }  return sum;  } | void main()  {  double e, x;  int k;  cout << "sin(x)/x = 1 - x^2/3! + x^4/5! - x^6/7! + ... where x from -infinity to +infinity (e=10^-k)" << endl;  cout << "Enter k: ";  cin >> k;  cout << "Enter x: ";  cin >> x;  cout << "using standart funtion: " << sinxDivx(x) << endl;  e = accuracy(k);  cout << "accuracy: " << e << endl;  cout << "answer: " << teylor(x, e) << endl;  } |

## Скриншоты:



# Использование цепочечных команд языка Ассемблер.

Задан текстовый файл Input.txt, состоящий из слов и некоторое предложение. Разделителями между словами является некоторое множество знаков препинания.

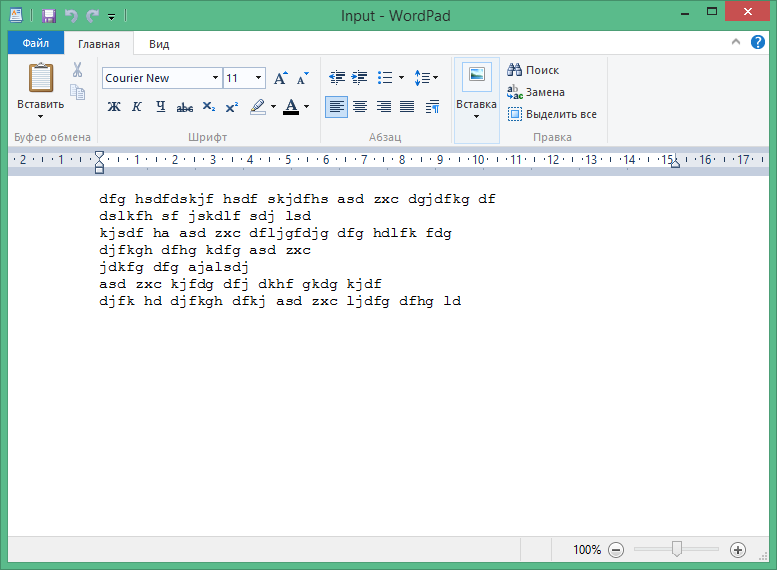
Найти строки, в которых встречается данное предложение, продублировать первое и последнее слова предложений. Результат записать в новый файл Output.txt.

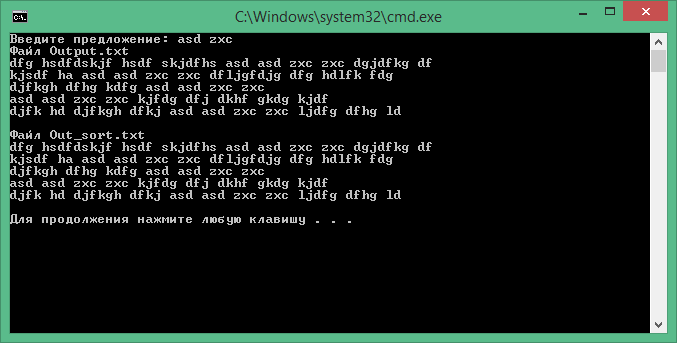
Упорядочить слова в искомых предложениях по убыванию длин слов, результат сортировки записать в файл Out\_sort.txt.

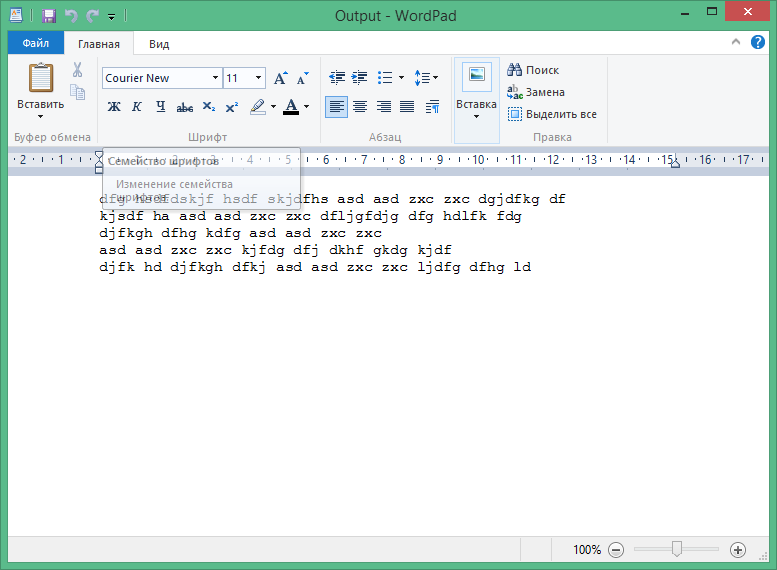
## Код:

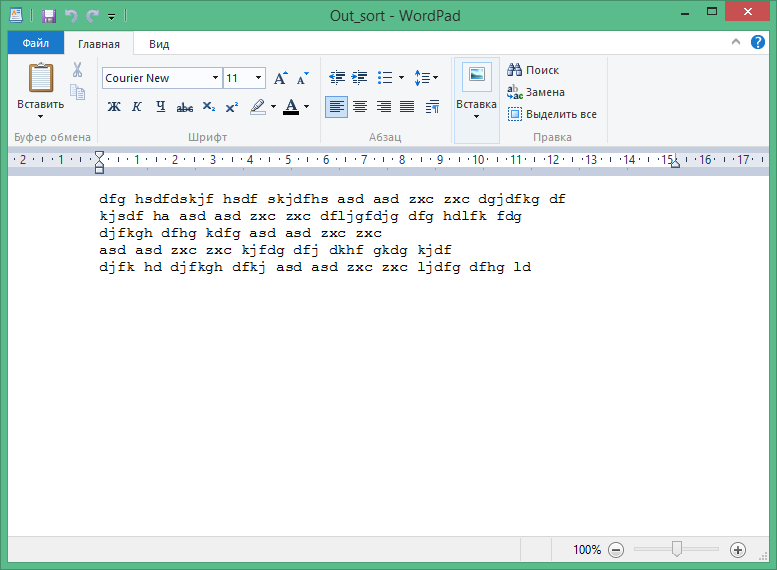
|  |  |  |
| --- | --- | --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include <cstring>  #include <stdio.h>  #include <fstream>  using namespace std;  bool fileCheck(ifstream &fin);  int asmstrlen(char\* str)  {  int len = 0;  \_asm  {  lea edx, [str]  lea eax, [edx - 1]  xor ecx, ecx  loop1 :  inc eax  cmp cl, [eax]  jnz loop1  sub eax, edx  mov len, eax  }  return len;  }  char\* asmstrcpy(char\* dest, char\* res)  {  int len = asmstrlen(res);  \_asm  {  mov ax, ds  mov es, ax  mov esi, res  mov edi, dest  A1 :  mov ax, [esi]  mov[edi], ax  add esi, 1  add edi, 1  cmp[esi], 0  jne A1  }  dest[asmstrlen(res)] = '\0';  return dest;  }  void counter(char \*acop, char \*tmp2, char \*delim, char \*tok2, int &count1)  {  while (tmp2)  {  count1++;  tmp2 = strtok\_s(NULL, delim, &tok2);  }  }  void tmpForm(char \*tmpfword, char \*tmps, char \*tmp1, char \*tmplword, char \*tmp2, int count1, int &count2)  {  count2++;  if (count2 == 1)  {  asmstrcpy(tmpfword, tmp2);  }  strcat(tmps, tmp1);  strcat(tmps, " ");  if (count2 == count1)  {  asmstrcpy(tmplword, tmp2);  }  }  void strForm(char \*tmpstr, char \*tmpfword, char \*tmplword, char \*tmps, char \*tmp1, int count1, int count2, bool &flag, bool &flag1)  {  if (count2 == count1)  {  flag = true;  flag1 = true;  strcat(tmpstr, tmpfword);  strcat(tmpstr, " ");  strcat(tmpstr, tmps);  strcat(tmpstr, tmplword);  strcat(tmpstr, " ");  if (tmp1 != 0)  {  strcat(tmpstr, tmp1);  strcat(tmpstr, " ");  }  }  else  {  strcat(tmpstr, tmps);  }  } | int output(char \*a, char \*delim)  {  char str[1000] = "", tmpstr[1200] = "", acop[100], tmps[200] = "", \*tmp1, \*tmp2, \*tok1 = NULL, \*tok2 = NULL, tmpfword[100] = "", tmplword[100] = "";  int count1 = 0, count2;  bool flag, flag1 = false;  ifstream fin("Input.txt");  ofstream fout("Output.txt");  strcpy\_s(acop, a);  tmp2 = strtok\_s(acop, delim, &tok2);  counter(acop, tmp2, delim, tok2, count1);  strcpy\_s(acop, a);  if (!fileCheck(fin))  {  fin.close();  fout.close();  return 0;  }  cout << "Файл Output.txt" << endl;  fin.clear();  fin.seekg(0);  while (fin)  {  flag = false;  fin.getline(str, 1000);  tmp1 = strtok\_s(str, delim, &tok1);  while (tmp1)  {  tmp2 = strtok\_s(acop, delim, &tok2);  count2 = 0;  if (tmp2 != 0 && strcmp(tmp1, tmp2) == 0)  {  while (tmp1 != 0 && tmp2 != 0)  {  if (strcmp(tmp1, tmp2) == 0)  {  tmpForm(tmpfword, tmps, tmp1, tmplword, tmp2, count1, count2);  }  else  {  break;  }  tmp1 = strtok\_s(NULL, delim, &tok1);  tmp2 = strtok\_s(NULL, delim, &tok2);  }  strForm(tmpstr, tmpfword, tmplword, tmps, tmp1, count1, count2, flag, flag1);  asmstrcpy(tmps, "");  }  else  {  strcat(tmpstr, tmp1);  strcat(tmpstr, " ");  }  asmstrcpy(acop, a);  tmp1 = strtok\_s(NULL, delim, &tok1);  }  if (flag)  {  fout << tmpstr << "\n";  cout << tmpstr << endl;  }  strcpy(tmpstr, "");  }  if (flag1 == false)  {  cout << "В данном файле нет искомого предложения!" << endl;  }  cout << endl;  fin.close();  fout.close();  return 1;  }  void strSort(int \*arrlen, char \*\*arrw, int i)  {  for (int a = 0; a <= i - 1; a++)  {  for (int b = 0; b <= (i - 1 - a); b++)  {  if (arrlen[b + 1] > arrlen[b])  {  swap(arrlen[b + 1], arrlen[b]);  swap(arrw[b + 1], arrw[b]);  }  }  }  } | void sortCounter(char \*str, char \*delim, int \*arrlen, char \*\*arrw, char \*word, int &i)  {  while (word)  {  arrlen[i] = asmstrlen(word);  arrw[i] = word;  word = strtok(NULL, delim);  i++;  }  }  int out\_sort(char \*delim)  {  char str[1200] = { 0 }, nstr[1200] = { 0 }, \*word;  char \*arrw[100] = { 0 };  int arrlen[100] = { 0 }, i;  ifstream finop("Output.txt");  ofstream fouts("Out\_sort.txt");  if (!fileCheck(finop))  {  finop.close();  fouts.close();  return 0;  }  cout << "Файл Out\_sort.txt" << endl;  finop.clear();  finop.seekg(0);  while (finop)  {  i = 0;  finop.getline(str, 1200);  word = strtok(str, delim);  sortCounter(str, delim, arrlen, arrw, word, i);  strSort(arrlen, arrw, i);  for (int a = 0; a <= i - 1; a++)  {  strcat(nstr, arrw[a]);  strcat(nstr, " ");  }  fouts << nstr << "\n";  cout << nstr;  strcpy(nstr, "");  cout << endl;  }  finop.close();  fouts.close();  return 1;  }  void main()  {  setlocale(LC\_ALL, "rus");  char a[100], delim[] = " ,./;:'[{}]()-\_=+!@#$%^&\*|\`~";  cout << "Введите предложение: ";  gets\_s(a);  output(a, delim);  out\_sort(delim);  }  bool fileCheck(ifstream &f)  {  if (!f.is\_open())  {  cout << "Файл не может быть открыт!" << endl;  f.close();  return false;  }  else  {  if (f.eof())  {  cout << "Входной файл пуст!" << endl;  f.close();  return false;  }  else  {  return true;  }  }  } |

## Скриншоты:









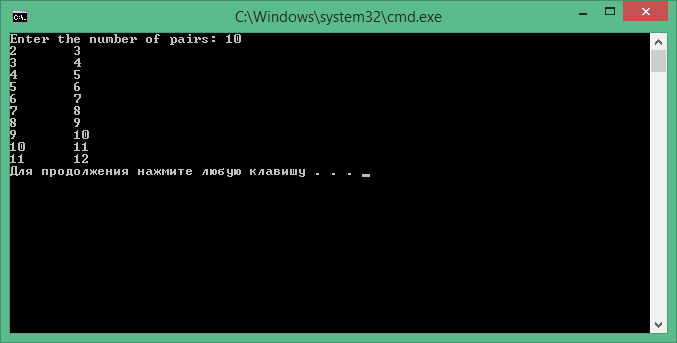
# Использование внешних подпрограмм языка Ассемблер.

Найти N пар взаимно-простых натуральных чисел.

## Код:

|  |  |
| --- | --- |
| **Файл Asmblr.asm**  .686  .model flat  .data  tmp dd 0  count dd 0  del dd 0  .code  Public \_func1  \_func1 proc  push ebp  mov ebp,esp  mov ecx, [ebp + 8]  mov esi, [ebp + 12]  inc ecx  mov ebx, 2  for\_ :  mov eax, ebx  mov [esi], eax  inc ebx  add esi, 4  loop for\_  pop ebp  ret  \_func1 endp  Public @func2@8  @func2@8 proc  ;mov edi, ecx  mov esi, edx  inc ecx  mov ebx, 2  for\_ :  mov eax, ebx  mov [esi], eax  inc ebx  add esi, 4  loop for\_  ret  @func2@8 endp  Public \_func3@8  \_func3@8 proc  push ebp  mov ebp,esp  mov ecx, [ebp + 8]  mov esi, [ebp + 12]  inc ecx  mov ebx, 2  for\_ :  mov eax, ebx  mov [esi], eax  inc ebx  add esi, 4  loop for\_  pop ebp  ret 8  \_func3@8 endp  end | **Файл Main.cpp**  #include <iostream>  using namespace std;  extern "C" void \_cdecl func1(int, int\*);  extern "C" void \_fastcall func2(int, int\*);  extern "C" void \_stdcall func3(int, int\*);  int main()  {  int size, \*arr;  cout << "Enter the number of pairs: ";  cin >> size;  arr = new int[size + 1];  func3(size, arr);  for (int i = 0; i < size; i++)  {  cout << arr[i] << "\t" << arr[i + 1] << endl;  }  delete[] arr;  return 0;  } |

## Скриншоты:



# WINAPI

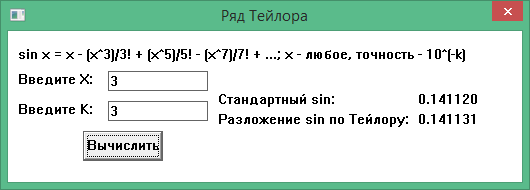
Вычислить ряд Тейлора. Создать оконное приложение.

sinx/x = 1 - x^2/3! + x^4/5! - x^6/7! + ... where x from -infinity to +infinity (e=10^-k)

**Код:**

|  |
| --- |
| #include <Windows.h>  #include <cmath>  #include <cstdio>  #define BTN 101  #define EDIT1 102  #define EDIT2 103  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  HWND hChildX, hChildK;  HINSTANCE hApp;  int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow)  {  static char szAppName[] = "UP4";  HWND hwnd;  MSG msg;  WNDCLASSEX wndclass;  wndclass.cbSize = sizeof (wndclass);  wndclass.style = CS\_HREDRAW | CS\_VREDRAW;  wndclass.lpfnWndProc = WndProc;  wndclass.cbClsExtra = 0;  wndclass.cbWndExtra = 0;  wndclass.hInstance = hInstance;  wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);  wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);  wndclass.hbrBackground = (HBRUSH)GetStockObject(DC\_BRUSH);  wndclass.lpszMenuName = NULL;  wndclass.lpszClassName = szAppName;  wndclass.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);  if (!RegisterClassEx(&wndclass))  {  MessageBox(NULL, "Невозможно зарегистрировать класс окна.", "Ошибка", MB\_OK);  return 0;  }  hwnd = CreateWindowEx(WS\_EX\_CONTROLPARENT, szAppName, "Ряд Тейлора", WS\_SYSMENU,  CW\_USEDEFAULT, 0, 530, 190,  NULL, NULL, hInstance, NULL);  if (!hwnd)  {  MessageBox(NULL, "Невозможно создать основное окно.", "Ошибка", MB\_OK);  return 0;  }  ShowWindow(hwnd, iCmdShow);  UpdateWindow(hwnd);  while (GetMessage(&msg, NULL, 0, 0))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return msg.wParam;  }  HWND editX;  HWND editK;  HWND button;  LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)  {  HDC hdc;  PAINTSTRUCT ps;  RECT rect;  LPSTR str = "sin x = x - (x^3)/3! + (x^5)/5! - (x^7)/7! + ...; x - любое, точность - 10^(-k)";  char strx[20], strk[20];  switch (iMsg)  {  case WM\_CREATE:  editX = CreateWindow("edit", "", WS\_CHILD | WS\_VISIBLE | WS\_BORDER | ES\_AUTOHSCROLL, 100, 40, 100, 20, hwnd,(HMENU) EDIT1, hApp, NULL); //WS\_TABSTOP  editK = CreateWindow("edit", "", WS\_CHILD | WS\_VISIBLE | WS\_BORDER | ES\_AUTOHSCROLL, 100, 70, 100, 20, hwnd,(HMENU) EDIT2, hApp, NULL);  button = CreateWindow("button", "Вычислить", WS\_CHILD | WS\_VISIBLE | WS\_BORDER | BS\_PUSHBUTTON, 75, 100, 80, 30, hwnd,(HMENU) BTN, hApp, NULL);  SetFocus(editX);  break;  case WM\_PAINT:  hdc = BeginPaint(hwnd, &ps);  TextOut(hdc, 10, 15, str, strlen(str));  TextOut(hdc, 10, 40, "Введите Х:", 10);  TextOut(hdc, 10, 70, "Введите K:", 10);  TextOut(hdc, 210, 60, "Стандартный sin:", 16);  TextOut(hdc, 210, 80, "Разложение sin по Тейлору:", 26);  EndPaint(hwnd, &ps);  break;  case WM\_COMMAND:  switch (wParam)  {  case BTN:  SendMessage(editX, WM\_GETTEXT, sizeof(strx) / sizeof(strx[0]), (LPARAM)strx);  SendMessage(editK, WM\_GETTEXT, sizeof(strk) / sizeof(strk[0]), (LPARAM)strk);  double x, e = 1, k, sum = 0, f = 1, y, b = 1, a = 0;  char stand[20], teyl[20];  x = atof(strx);  k = atof(strk);  y = sin(x);  for (int i = 0; i < k; i++)  {  e /= 10;  }  do  {  a++;  f \*= a;  b \*= x;  sum += (b / f);  a++;  f \*= a;  b \*= -x;  } while (abs(b / f) >= e);  sprintf\_s(stand, "%f", y);  sprintf\_s(teyl, "%f", sum);  hdc = GetDC(hwnd);  SetTextAlign(hdc, TA\_LEFT | TA\_TOP);  //SetBkMode(hdc, TRANSPARENT);  TextOut(hdc, 410, 60, stand, 8);  TextOut(hdc, 410, 80, teyl, 8);  ReleaseDC(hwnd, hdc);  break;  }  break;  case WM\_CLOSE:  DestroyWindow(hwnd);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hwnd, iMsg, wParam, lParam);  }  return DefWindowProc(hwnd, iMsg, wParam, lParam);  } |

## Скриншоты:



# Работа с графикой WINAPI

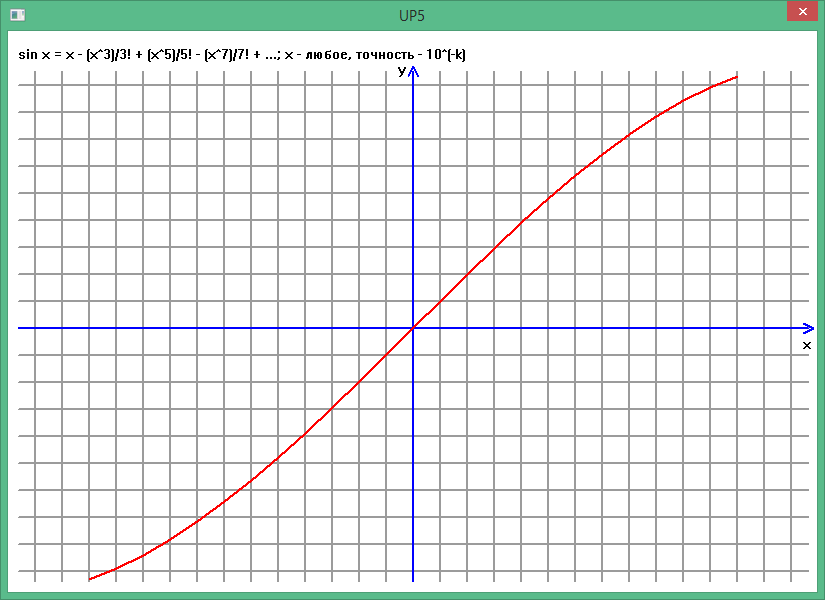
Вычислить ряд Тейлора. Создать оконное приложение с графическим интерфейсом.

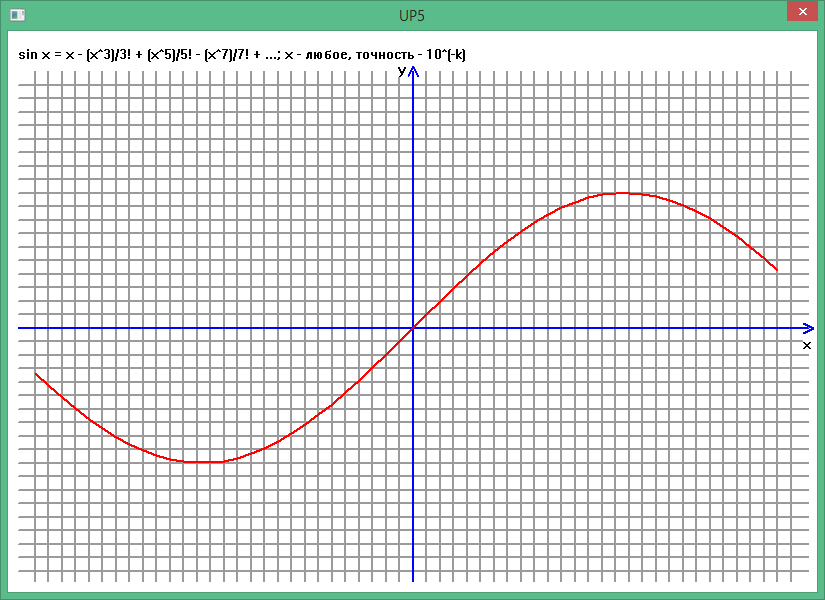
sinx/x = 1 - x^2/3! + x^4/5! - x^6/7! + ... where x from -infinity to +infinity (e=10^-k)

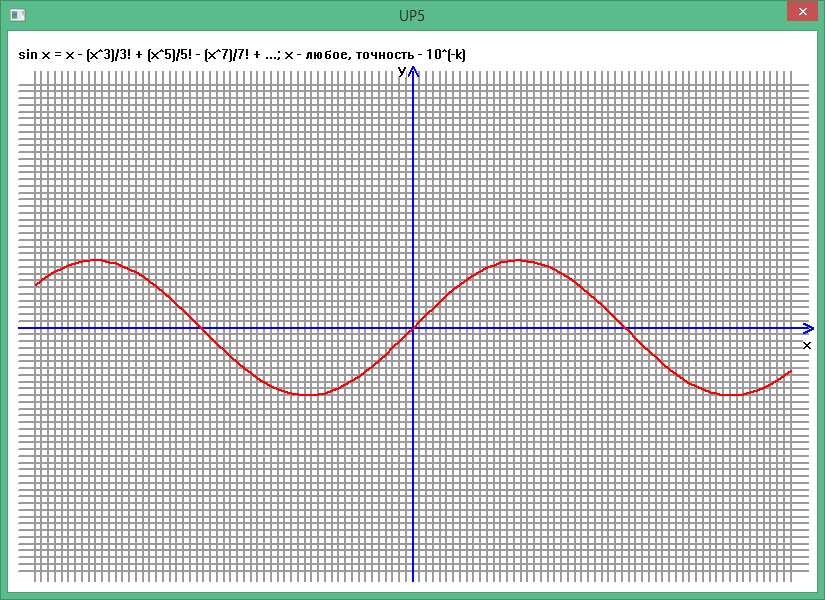
## Код:

|  |
| --- |
| #include <Windows.h>  #include <cmath>  #include <cstdio>  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  void Coord(HDC);  void Teyl(HDC);  int ed = 27;  double koef = 1;  int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)  {  HWND hwnd;  static char szClassName[] = "UP5";  MSG msg;  WNDCLASSEX wndclass;  wndclass.cbSize = sizeof (wndclass);  wndclass.style = CS\_HREDRAW | CS\_VREDRAW;  wndclass.lpfnWndProc = WndProc;  wndclass.cbClsExtra = 0;  wndclass.cbWndExtra = 0;  wndclass.hInstance = hInstance;  wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);  wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);  wndclass.hbrBackground = (HBRUSH)GetStockObject(DC\_BRUSH);  wndclass.lpszMenuName = NULL;  wndclass.lpszClassName = szClassName;  wndclass.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);  if (!RegisterClassEx(&wndclass))  {  MessageBox(NULL, "Невозможно зарегистрировать класс окна.", "Ошибка", MB\_OK);  return 0;  }  hwnd = CreateWindowEx(WS\_EX\_CONTROLPARENT, szClassName, "UP5", WS\_SYSMENU,  CW\_USEDEFAULT, 0, 825, 600, NULL, NULL, hInstance, NULL);  if (!hwnd)  {  MessageBox(NULL, "Невозможно создать основное окно.", "Ошибка", MB\_OK);  return 0;  }  ShowWindow(hwnd, nCmdShow);  UpdateWindow(hwnd);  while (GetMessage(&msg, NULL, 0, 0))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return msg.wParam;  }  LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)  {  HDC hdc;  PAINTSTRUCT ps;  RECT rect;  LPSTR str = "sin x = x - (x^3)/3! + (x^5)/5! - (x^7)/7! + ...; x - любое, точность - 10^(-k)";  switch (iMsg)  {  /\*case WM\_CREATE:  break;\*/  case WM\_PAINT:  hdc = BeginPaint(hwnd, &ps);  TextOut(hdc, 10, 15, str, strlen(str));  Coord(hdc);  Teyl(hdc);  EndPaint(hwnd, &ps);  break;  case WM\_KEYDOWN:  switch (wParam)  {  case VK\_ADD:  if (koef < 4)  {  koef \*= 2;  InvalidateRect(hwnd, 0, TRUE);  }  break;  case VK\_SUBTRACT:  if (koef > 0.25)  {  koef /= 2;  InvalidateRect(hwnd, 0, TRUE);  }  break;  }  case WM\_MOUSEWHEEL:  if (GET\_WHEEL\_DELTA\_WPARAM(wParam) > 0) //прокрутка колёсика мыши вперёд  {  if (koef < 4)  {  koef \*= 2;  InvalidateRect(hwnd, 0, TRUE);  }  }  if (GET\_WHEEL\_DELTA\_WPARAM(wParam) < 0)  {  if (koef > 0.25)  {  koef /= 2;  InvalidateRect(hwnd, 0, TRUE);  }  }  break;  case WM\_CLOSE:  DestroyWindow(hwnd);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hwnd, iMsg, wParam, lParam);  }  return DefWindowProc(hwnd, iMsg, wParam, lParam);  }  void Coord(HDC hdc)  {  HPEN pen1 = CreatePen(PS\_SOLID, 2, RGB(155, 155, 155));  for (int i = 2 \* koef; i <= 20 \* koef; i++) // разметка  {  SelectObject(hdc, pen1);  MoveToEx(hdc, 10, i \* ed / koef, NULL);  LineTo(hdc, 800, i \* ed / koef);  }  for (int i = 1 \* koef; i <= 29 \* koef; i++)  {  SelectObject(hdc, pen1);  MoveToEx(hdc, i \* ed / koef, 40, NULL);  LineTo(hdc, i \* ed / koef, 550);  }  DeleteObject(pen1);  HPEN pen2 = CreatePen(PS\_SOLID, 2, RGB(0, 0, 255));  SelectObject(hdc, pen2);  MoveToEx(hdc, 800, 297, NULL); // Ox  LineTo(hdc, 10, 297);  MoveToEx(hdc, 405, 40, NULL); // Oy  LineTo(hdc, 405, 550);  MoveToEx(hdc, 805, 297, NULL); // стрелочка Ox  LineTo(hdc, 795, 292);  MoveToEx(hdc, 805, 297, NULL);  LineTo(hdc, 795, 302);  MoveToEx(hdc, 405, 35, NULL); // стрелочка Oy  LineTo(hdc, 400, 45);  MoveToEx(hdc, 405, 35, NULL);  LineTo(hdc, 410, 45);  DeleteObject(pen2);  TextOut(hdc, 390, 30, "y", 1);  TextOut(hdc, 795, 305, "x", 1);  }  void Teyl(HDC hdc)  {  HPEN pen = CreatePen(PS\_SOLID, 2, RGB(255, 0, 0));  double x, e = 0.001, sum, fuct, s, xs, y;  double i = koef;  bool flag = true;  SelectObject(hdc, pen);  for (x = -1.4 \* koef; x <= 1.4 \* koef; x += 0.1)  {  sum = 0;  fuct = 1;  s = 0;  xs = 1;  do  {  s++;  fuct \*= s;  xs \*= x;  sum += (xs / fuct);  s++;  fuct \*= s;  xs \*= -x;  } while (abs(xs / fuct) >= e);  y = 297 - sum \* ed \* 10 / koef;  if (y > 40 && y < 550)  {  if (flag)  {  MoveToEx(hdc, i \* ed / koef, y, NULL);  flag = false;  }  else  {  LineTo(hdc, i \* ed / koef, y);  }  }  i++;  }  DeleteObject(pen);  } |

## Скриншоты:







# Создание приложения WINAPI

Объявить абстрактный базовый класс «Фигура на плоскости», моделирующий общие свойства и поведение геометрических фигур. Включить в объявления классов следующее поля данных:

– название фигуры (отрезок, квадрат, круг и т.д.);

­– одно или несколько полей типа *Point*, характеризующие положение фигуры на плоскости. Например: центр круга (одно поле данных) или два поля для прямоугольника и квадрата (верхний левый и правый нижний углы фигуры);

– поля, характеризующие параметры фигуры: количество сторон многоугольника, периметр и площадь фигуры.

В класс наследник включить следующие методы:

– конструктор по умолчанию, конструктор с формальными параметрами и конструктор копирования, а также виртуальный деструктор;

– виртуальные методы для вычисления параметров объекта (периметра и площади фигуры с учетом ее вида; площади поверхностей и объем тела);

– виртуальный метод для отображения объекта на экране. В соответствующей функции-члене класса достаточно вывести текст «Рисунок <name> <координаты> », где name - название фигуры.

– виртуальный метод для моделирования перемещения фигуры на плоскости (сдвиг центра окружности или вершин четырехугольника); В соответствующей функции-члене класса достаточно выводить текст «Движение <name> <координаты> », где name - название фигуры.

– виртуальный метод для моделирования сжатия или растяжения фигуры (увеличение радиуса круга, уменьшение стороны квадрата).

В приведенных формулах *S* – площадь, Р (или р ) – периметр фигуры.

## Код:

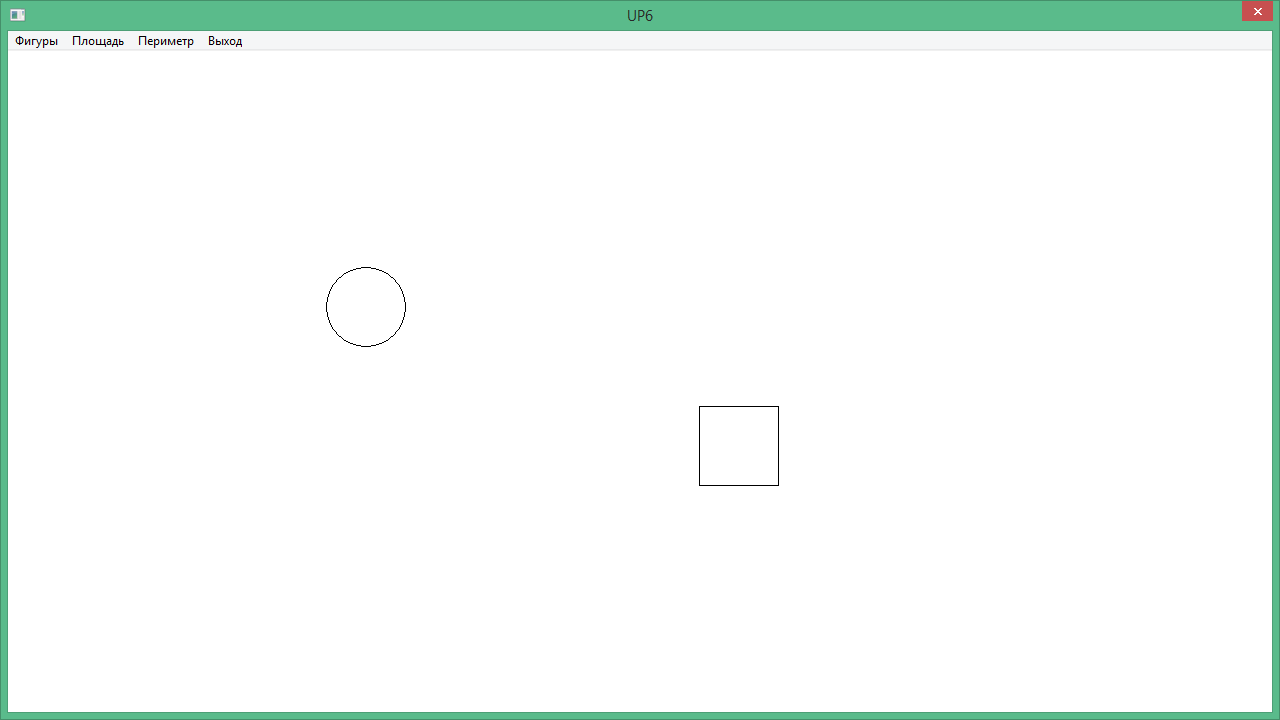
|  |
| --- |
| **Файл Figure.h**  #ifndef FIGURE\_H  #define FIGURE\_H  #define PI 3.14159265  #include <iostream>  #include <Windows.h>  #include <cmath>  using namespace std;  class Figure  {  protected:  char\* name;  int lPoint[2];  int rPoint[2];  int side;  double perim;  double square;  public:  virtual void fPerimeter() = 0;  virtual void fSquare() = 0;  virtual void fMove(int x, int y) = 0;  virtual void fAlteration(int x, int y) = 0;  virtual void fDraw(HDC hdc) = 0;  };  #endif |

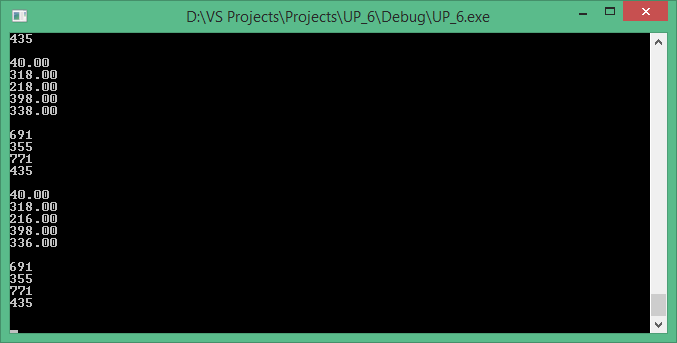
|  |  |
| --- | --- |
| **Файл Circle.h**  #ifndef CIRCLE\_H  #define CIRCLE\_H  #include "Figure.h"  class Circle : public Figure  {  double R;  public:  Circle();  Circle(int lx, int ly, int rx, int ry);  Circle(const Circle& init);  virtual ~Circle();  void cRadius();  int getLXY(int index);  int getRXY(int index);  double getPerimeter();  double getSquare();  double getR();  virtual void fPerimeter();  virtual void fSquare();  virtual void fMove(int x, int y);  virtual void fAlteration(int x, int y);  virtual void fDraw(HDC hdc);  };  #endif | **Файл Circle.cpp**  #include "Circle.h"  Circle::Circle()  {  name = new char[strlen("Круг") + 1];  strcpy(name, "Круг");  for (int i = 0; i < 2; i++)  {  lPoint[i] = 0;  rPoint[i] = 0;  }  side = 1;  }  Circle::Circle(int lx, int ly, int rx, int ry)  {  name = new char[strlen("Круг") + 1];  strcpy(name, "Круг");  lPoint[0] = lx;  lPoint[1] = ly;  rPoint[0] = rx;  rPoint[1] = ry;  cRadius();  fPerimeter();  fSquare();  side = 1;  }  Circle::Circle(const Circle& init)  {  name = new char[strlen("Круг") + 1];  strcpy(name, "Круг");  for (int i = 0; i < 2; i++)  {  lPoint[i] = init.lPoint[i];  rPoint[i] = init.rPoint[i];  }  cRadius();  perim = getPerimeter();  square = getSquare();  side = 1;  }  Circle::~Circle()  {  delete[] name;  }  void Circle::cRadius()  {  R = sqrt((rPoint[0] - lPoint[0])\*(rPoint[0] - lPoint[0]) + (rPoint[1] - lPoint[1])\*(rPoint[1] - lPoint[1]));  }  int Circle::getLXY(int index)  {  if (index >= 0 && index <= 1)  {  return lPoint[index];  }  return 0;  }  int Circle::getRXY(int index)  {  if (index >= 0 && index <= 1)  {  return rPoint[index];  }  return 0;  }  double Circle::getPerimeter()  {  return perim;  }  double Circle::getSquare()  {  return square;  }  double Circle::getR()  {  return R;  }  void Circle::fPerimeter()  {  perim = 2 \* PI \* R;  }  void Circle::fSquare()  {  square = PI \* R \* R;  }  void Circle::fMove(int x, int y)  {  lPoint[0] += x;  lPoint[1] += y;  rPoint[0] += x;  rPoint[1] += y;  }  void Circle::fAlteration(int x, int y)  {  rPoint[0] += x;  rPoint[1] += y;  fPerimeter();  fSquare();  cRadius();  }  void Circle::fDraw(HDC hdc)  {  cRadius();  printf("%.2f\n", R);  printf("%.2f\n", lPoint[0] - R);  printf("%.2f\n", lPoint[1] - R);  printf("%.2f\n", rPoint[0] + R);  printf("%.2f\n\n", rPoint[1] + R);  Ellipse(hdc, lPoint[0] - R, lPoint[1] + R, lPoint[0] + R, lPoint[1] - R);  } |

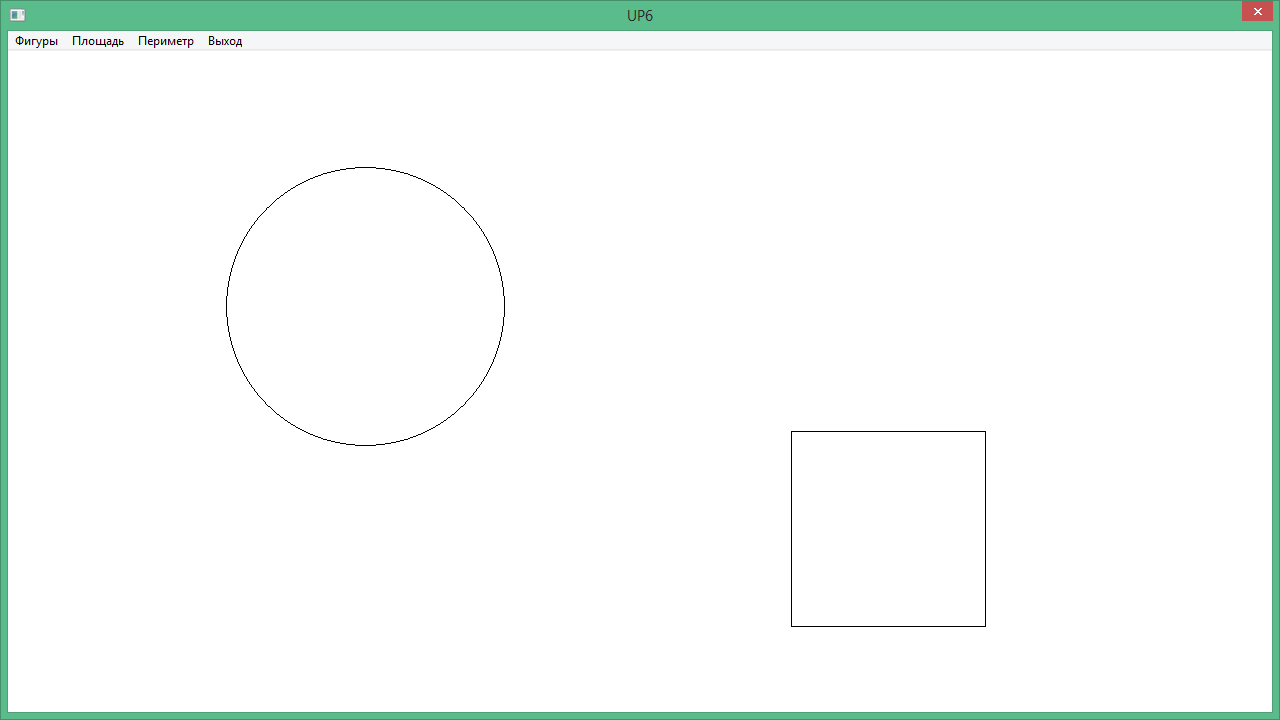
|  |  |
| --- | --- |
| **Файл Rectangle.h**  #ifndef RECTANGLE\_H  #define RECTANGLE\_H  #include "Figure.h"  class Rectangles : public Figure  {  public:  Rectangles();  Rectangles(int lx, int ly, int rx, int ry);  Rectangles(const Rectangles& init);  virtual ~Rectangles();  int getLXY(int index);  int getRXY(int index);  double getPerimeter();  double getSquare();  virtual void fPerimeter();  virtual void fSquare();  virtual void fMove(int x, int y);  virtual void fAlteration(int x, int y);  virtual void fDraw(HDC hdc);  };  #endif | **Файл Rectangle.cpp**  #include "Rectangle.h"  Rectangles::Rectangles()  {  name = new char[strlen("Прямоугольник") + 1];  strcpy(name, "Прямоугольник");  for (int i = 0; i < 2; i++)  {  lPoint[i] = 0;  rPoint[i] = 0;  }  side = 4;  }  Rectangles::Rectangles(int lx, int ly, int rx, int ry)  {  name = new char[strlen("Прямоугольник") + 1];  strcpy(name, "Прямоугольник");  lPoint[0] = lx;  lPoint[1] = ly;  rPoint[0] = rx;  rPoint[1] = ry;  perim = getPerimeter();  square = getSquare();  side = 4;  }  Rectangles::Rectangles(const Rectangles& init)  {  name = new char[strlen("Прямоугольник") + 1];  strcpy(name, "Прямоугольник");  for (int i = 0; i < 2; i++)  {  lPoint[i] = init.lPoint[i];  rPoint[i] = init.rPoint[i];  }  fPerimeter();  fSquare();  side = init.side;  }  Rectangles::~Rectangles()  {  delete[] name;  }  int Rectangles::getLXY(int index)  {  if (index >= 0 && index <= 1)  {  return lPoint[index];  }  return 0;  }  int Rectangles::getRXY(int index)  {  if (index >= 0 && index <= 1)  {  return rPoint[index];  }  return 0;  }  double Rectangles::getPerimeter()  {  return perim;  }  double Rectangles::getSquare()  {  return square;  }  void Rectangles::fPerimeter()  {  perim = 2 \* ((rPoint[0] - lPoint[0]) + (rPoint[1] - lPoint[1]));  }  void Rectangles::fSquare()  {  square = (rPoint[0] - lPoint[0]) \* (rPoint[1] - lPoint[1]);  }  void Rectangles::fMove(int x, int y)  {  lPoint[0] += x;  lPoint[1] += y;  rPoint[0] += x;  rPoint[1] += y;  }  void Rectangles::fAlteration(int x, int y)  {  rPoint[0] += x;  rPoint[1] += y;  fPerimeter();  fSquare();  }  void Rectangles::fDraw(HDC hdc)  {  printf("%.2d\n", lPoint[0]);  printf("%.2d\n", lPoint[1]);  printf("%.2d\n", rPoint[0]);  printf("%.2d\n\n", rPoint[1]);  Rectangle(hdc, lPoint[0], lPoint[1], rPoint[0], rPoint[1]);  } |

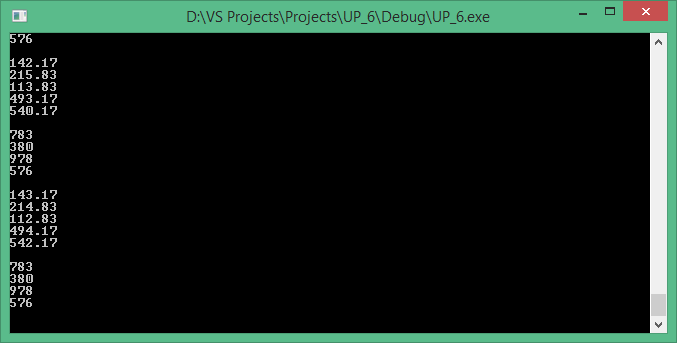
|  |
| --- |
| **Файл Main.cpp**  #include "Windows.h"  #include "cstdio"  #include "Circle.h"  #include "Rectangle.h"  #include <io.h> // для \_open\_osfhandle  #include <fcntl.h> // для \_O\_TEXT  LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam);  Circle circ(640, 360, 640, 400);  Rectangles rectan(600, 320, 680, 400);  bool cflag = false, rflag = false, cActive = false, rActive = false;  int i = 0;  double temp;  char out[20];  BOOL CreateConsole(void) // создание консоли  {  FreeConsole(); // отключает все процессы от данной консоли  if (AllocConsole()) // назначает новую консоль для данного процесса  {  int hCrt = \_open\_osfhandle((long) // возвращает идентификатор файла времени выполнения  GetStdHandle(STD\_OUTPUT\_HANDLE), \_O\_TEXT); /\* извлекает дескриптор для стандартного ввода данных, стандартного вывода или  стандартной ошибки устройства\*/  \*stdout = \*(::\_fdopen(hCrt, "w")); // перенаправление стандартного потока вывода | \_fdopen // ассоциация потока с файлом  ::setvbuf(stdout, NULL, \_IONBF, 0);  \*stderr = \*(::\_fdopen(hCrt, "w")); // перенаправление стандартного потока ошибок  ::setvbuf(stderr, NULL, \_IONBF, 0);  return TRUE;  }return FALSE;  }  int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)  {  HWND hwnd;  static char szClassName[] = "UP6";  MSG msg;  WNDCLASSEX wndclass;  HMENU hMenu;  HMENU hPopUpMenu;  HMENU hPopUpPerim;  HMENU hPopUpSquare;  wndclass.cbSize = sizeof (wndclass);  wndclass.style = CS\_HREDRAW | CS\_VREDRAW;  wndclass.lpfnWndProc = WndProc;  wndclass.cbClsExtra = 0;  wndclass.cbWndExtra = 0;  wndclass.hInstance = hInstance;  wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);  wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);  wndclass.hbrBackground = (HBRUSH)GetStockObject(DC\_BRUSH);  wndclass.lpszMenuName = NULL;  wndclass.lpszClassName = szClassName;  wndclass.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);  if (!RegisterClassEx(&wndclass))  {  MessageBox(NULL, "Невозможно зарегистрировать класс окна.", "Ошибка", MB\_OK);  return 0;  }  hwnd = CreateWindowEx(WS\_EX\_CONTROLPARENT, szClassName, "UP6", WS\_SYSMENU,  150, 100, 1280, 720, NULL, NULL, hInstance, NULL);  if (!hwnd)  {  MessageBox(NULL, "Невозможно создать основное окно.", "Ошибка", MB\_OK);  return 0;  }  hMenu = CreateMenu();  hPopUpMenu = CreateMenu();  hPopUpPerim = CreateMenu();  hPopUpSquare = CreateMenu();  AppendMenu(hMenu, MF\_STRING | MF\_POPUP, (UINT)hPopUpMenu, "Фигуры");  AppendMenu(hPopUpMenu, MF\_STRING, 101, "Круг");  AppendMenu(hPopUpMenu, MF\_STRING, 102, "Прямоугольник");  AppendMenu(hPopUpMenu, MF\_STRING, 103, "Очистить");  AppendMenu(hMenu, MF\_STRING | MF\_POPUP, (UINT)hPopUpSquare, "Площадь");  AppendMenu(hPopUpSquare, MF\_STRING, 104, "Круг");  AppendMenu(hPopUpSquare, MF\_STRING, 105, "Прямоугольник");  AppendMenu(hMenu, MF\_STRING | MF\_POPUP, (UINT)hPopUpPerim, "Периметр");  AppendMenu(hPopUpPerim, MF\_STRING, 106, "Круг");  AppendMenu(hPopUpPerim, MF\_STRING, 107, "Прямоугольник");  AppendMenu(hMenu, MF\_STRING, 108, "Выход");  SetMenu(hwnd, hMenu);  ShowWindow(hwnd, nCmdShow);  UpdateWindow(hwnd);  CreateConsole(); // создание консоли  while (GetMessage(&msg, NULL, 0, 0))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return msg.wParam;  }  LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)  {  HDC hdc, hdcMem;  PAINTSTRUCT ps;  RECT rect;  switch (iMsg)  {  case WM\_CREATE:  break;  case WM\_PAINT:  hdc = BeginPaint(hwnd, &ps);  if (cflag)  {  circ.fDraw(hdc);  }  if (rflag)  {  rectan.fDraw(hdc);  }  EndPaint(hwnd, &ps);  break;  case WM\_COMMAND:  switch (wParam)  {  case 101:  hdc = GetDC(hwnd);  circ.fDraw(hdc);  cflag = true;  ReleaseDC(hwnd, hdc);  break;  case 102:  hdc = GetDC(hwnd);  rectan.fDraw(hdc);  rflag = true;  ReleaseDC(hwnd, hdc);  break;  case 103:  cflag = false;  rflag = false;  InvalidateRect(hwnd, 0, TRUE);  break;  case 104:  temp = circ.getSquare();  sprintf\_s(out, "%f", temp);  MessageBox(hwnd, out, "Площадь круга", 0);  break;  case 105:  temp = rectan.getSquare();  sprintf\_s(out, "%f", temp);  MessageBox(hwnd, out, "Площадь прямоугольник", 0);  break;  case 106:  temp = circ.getPerimeter();  sprintf\_s(out, "%f", temp);  MessageBox(hwnd, out, "Периметр круг", 0);  break;  case 107:  temp = rectan.getPerimeter();  sprintf\_s(out, "%f", temp);  MessageBox(hwnd, out, "Периметр прямоугольника", 0);  break;  case 108:  DestroyWindow(hwnd);  break;  }  break;  case WM\_LBUTTONDOWN:  if (cflag && abs(LOWORD(lParam) - circ.getLXY(0)) <= circ.getR() && abs(HIWORD(lParam) - circ.getLXY(1)) <= circ.getR())  {  cActive = true;  }  if (rflag && LOWORD(lParam) >= rectan.getLXY(0) && HIWORD(lParam) >= rectan.getLXY(1)  && LOWORD(lParam) <= rectan.getRXY(0) && HIWORD(lParam) <= rectan.getRXY(1))  {  rActive = true;  }  break;  case WM\_MOUSEMOVE:  if (wParam & MK\_SHIFT && (abs(LOWORD(lParam) - circ.getRXY(0)) < 20 && abs(HIWORD(lParam) - circ.getRXY(1)) < 20))  {  SetCursor(LoadCursor(NULL, IDC\_SIZENS));  ShowCursor(TRUE);  }  if (wParam & MK\_SHIFT && (abs(LOWORD(lParam) - rectan.getRXY(0)) < 20 && abs(HIWORD(lParam) - rectan.getRXY(1)) < 20))  {  SetCursor(LoadCursor(NULL, IDC\_SIZENWSE));  ShowCursor(TRUE);  }  if (wParam & MK\_LBUTTON)  {  if (wParam & MK\_SHIFT)  {  if (cActive && abs(LOWORD(lParam) - circ.getRXY(0)) < 20 && abs(HIWORD(lParam) - circ.getRXY(1)) < 20)  {  circ.fAlteration(LOWORD(lParam) - circ.getRXY(0), HIWORD(lParam) - circ.getRXY(1));  InvalidateRect(hwnd, 0, TRUE);  }  if (rActive && abs(LOWORD(lParam) - rectan.getRXY(0)) < 20 && abs(HIWORD(lParam) - rectan.getRXY(1)) < 20)  {  rectan.fAlteration(LOWORD(lParam) - rectan.getRXY(0), HIWORD(lParam) - rectan.getRXY(1));  InvalidateRect(hwnd, 0, TRUE);  }  }  else  {  if (cActive)  {  circ.fMove(LOWORD(lParam) - circ.getLXY(0), HIWORD(lParam) - circ.getLXY(1));  InvalidateRect(hwnd, 0, TRUE);  }  if (rActive)  {  rectan.fMove(LOWORD(lParam) - rectan.getLXY(0), HIWORD(lParam) - rectan.getLXY(1));  InvalidateRect(hwnd, 0, TRUE);  }  }  }  break;  case WM\_LBUTTONUP:  cActive = false;  rActive = false;  break;  case WM\_CLOSE:  DestroyWindow(hwnd);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hwnd, iMsg, wParam, lParam);  }  return DefWindowProc(hwnd, iMsg, wParam, lParam);  } |

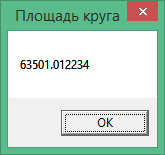
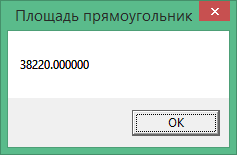
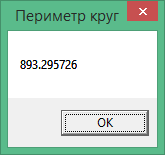
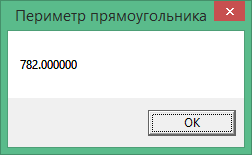
## Скриншоты:









# Создание приложения Windows Forms

Объявить абстрактный базовый класс «Фигура на плоскости», моделирующий общие свойства и поведение геометрических фигур. Включить в объявления классов следующее поля данных:

– название фигуры (отрезок, квадрат, круг и т.д.);

­– одно или несколько полей типа *Point*, характеризующие положение фигуры на плоскости. Например: центр круга (одно поле данных) или два поля для прямоугольника и квадрата (верхний левый и правый нижний углы фигуры);

– поля, характеризующие параметры фигуры: количество сторон многоугольника, периметр и площадь фигуры.

В класс наследник включить следующие методы:

– конструктор по умолчанию, конструктор с формальными параметрами и конструктор копирования, а также виртуальный деструктор;

– виртуальные методы для вычисления параметров объекта (периметра и площади фигуры с учетом ее вида; площади поверхностей и объем тела);

– виртуальный метод для отображения объекта на экране. В соответствующей функции-члене класса достаточно вывести текст «Рисунок <name> <координаты> », где name - название фигуры.

– виртуальный метод для моделирования перемещения фигуры на плоскости (сдвиг центра окружности или вершин четырехугольника); В соответствующей функции-члене класса достаточно выводить текст «Движение <name> <координаты> », где name - название фигуры.

– виртуальный метод для моделирования сжатия или растяжения фигуры (увеличение радиуса круга, уменьшение стороны квадрата).

## Код:

|  |
| --- |
| **Файл Figure.h**  #ifndef FIGURE\_H  #define FIGURE\_H  #define PI 3.14159265  #include <iostream>  #include <Windows.h>  #include <cmath>  using namespace std;  class Figure  {  protected:  char\* name;  int lPoint[2];  int rPoint[2];  int side;  double perim;  double square;  public:  virtual void fPerimeter() = 0;  virtual void fSquare() = 0;  virtual void fMove(int x, int y) = 0;  virtual void fAlteration(int x, int y) = 0;  };  #endif |

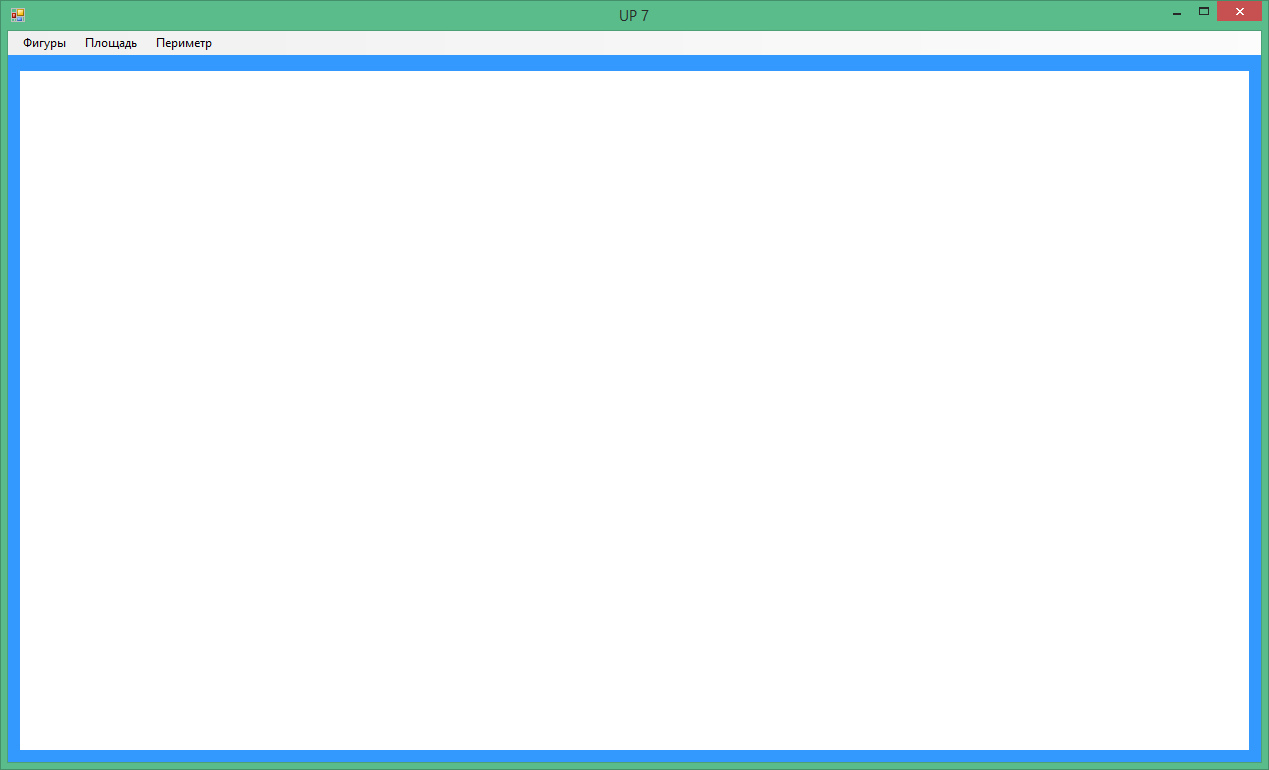
|  |  |
| --- | --- |
| **Файл Circle.h**  #ifndef CIRCLE\_H  #define CIRCLE\_H  #include "Figure.h"  class Circle : public Figure  {  double R;  public:  Circle();  Circle(int lx, int ly, int rx, int ry);  Circle(const Circle& init);  virtual ~Circle();  void cRadius();  int getLXY(int index);  int getRXY(int index);  double getPerimeter();  double getSquare();  double getR();  virtual void fPerimeter();  virtual void fSquare();  virtual void fMove(int x, int y);  virtual void fAlteration(int x, int y);  };  #endif | **Файл Circle.cpp**  #include "Circle.h"  Circle::Circle()  {  name = new char[strlen("Круг") + 1];  strcpy(name, "Круг");  for (int i = 0; i < 2; i++)  {  lPoint[i] = 0;  rPoint[i] = 0;  }  side = 1;  }  Circle::Circle(int lx, int ly, int rx, int ry)  {  name = new char[strlen("Круг") + 1];  strcpy(name, "Круг");  lPoint[0] = lx;  lPoint[1] = ly;  rPoint[0] = rx;  rPoint[1] = ry;  cRadius();  fPerimeter();  fSquare();  side = 1;  }  Circle::Circle(const Circle& init)  {  name = new char[strlen("Круг") + 1];  strcpy(name, "Круг");  for (int i = 0; i < 2; i++)  {  lPoint[i] = init.lPoint[i];  rPoint[i] = init.rPoint[i];  }  R = init.R;  side = 1;  }  Circle::~Circle()  {  delete[] name;  }  void Circle::cRadius()  {  R = sqrt((rPoint[0] - lPoint[0])\*(rPoint[0] - lPoint[0]) + (rPoint[1] - lPoint[1])\*(rPoint[1] - lPoint[1]));  }  int Circle::getLXY(int index)  {  if (index >= 0 && index <= 1)  {  return lPoint[index];  }  return 0;  }  int Circle::getRXY(int index)  {  if (index >= 0 && index <= 1)  {  return rPoint[index];  }  return 0;  }  double Circle::getPerimeter()  {  return perim;  }  double Circle::getSquare()  {  return square;  }  double Circle::getR()  {  return R;  }  void Circle::fPerimeter()  {  perim = 2 \* PI \* R;  }  void Circle::fSquare()  {  square = PI \* R \* R;  }  void Circle::fMove(int x, int y)  {  lPoint[0] += x;  lPoint[1] += y;  rPoint[0] += x;  rPoint[1] += y;  }  void Circle::fAlteration(int x, int y)  {  rPoint[0] += x;  rPoint[1] += y;  fPerimeter();  fSquare();  cRadius();  } |

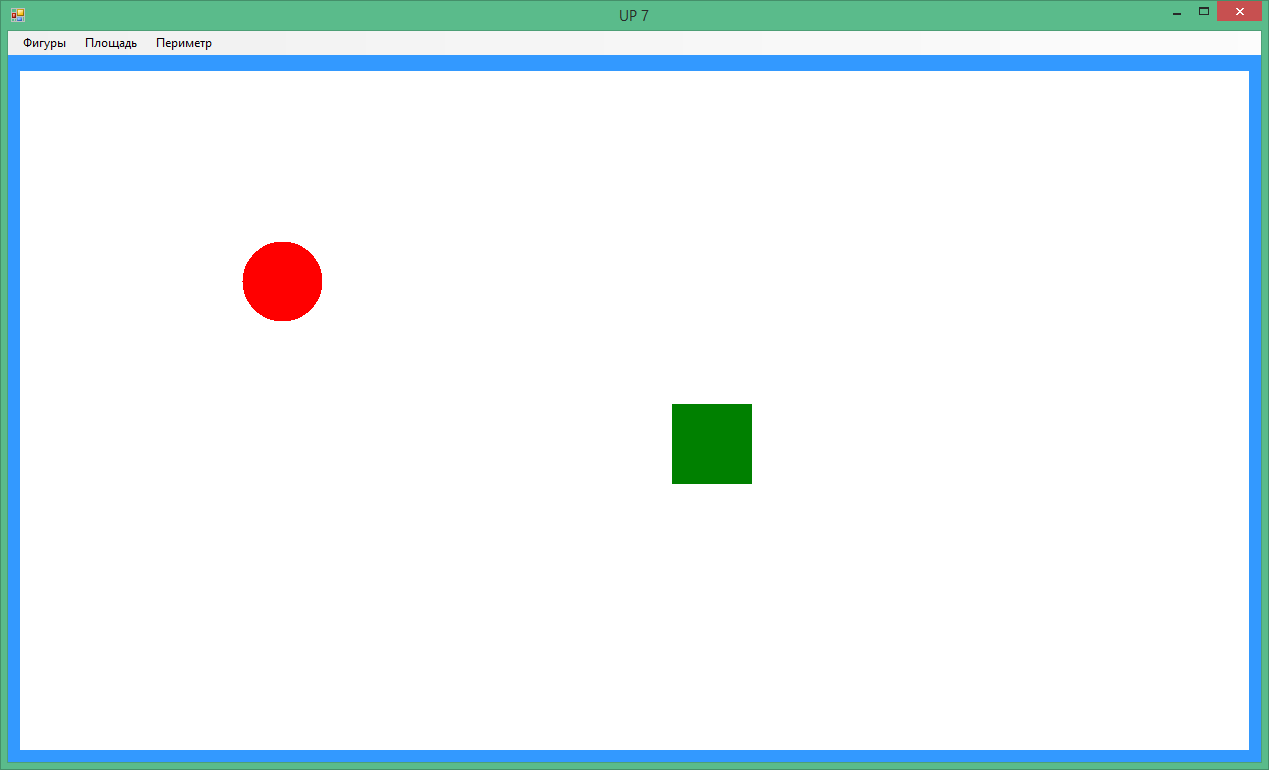
|  |  |
| --- | --- |
| **Файл Rectangle.h**  #ifndef RECTANGLE\_H  #define RECTANGLE\_H  #include "Figure.h"  class Rectangles : public Figure  {  public:  Rectangles();  Rectangles(int lx, int ly, int rx, int ry);  Rectangles(const Rectangles& init);  virtual ~Rectangles();  int getLXY(int index);  int getRXY(int index);  double getPerimeter();  double getSquare();  virtual void fPerimeter();  virtual void fSquare();  virtual void fMove(int x, int y);  virtual void fAlteration(int x, int y);  };  #endif | **Файл Rectangle.cpp**  #include "Rectangle.h"  Rectangles::Rectangles()  {  name = new char[strlen("Прямоугольник") + 1];  strcpy(name, "Прямоугольник");  for (int i = 0; i < 2; i++)  {  lPoint[i] = 0;  rPoint[i] = 0;  }  side = 4;  }  Rectangles::Rectangles(int lx, int ly, int rx, int ry)  {  name = new char[strlen("Прямоугольник") + 1];  strcpy(name, "Прямоугольник");  lPoint[0] = lx;  lPoint[1] = ly;  rPoint[0] = rx;  rPoint[1] = ry;  fPerimeter();  fSquare();  side = 4;  }  Rectangles::Rectangles(const Rectangles& init)  {  name = new char[strlen("Прямоугольник") + 1];  strcpy(name, "Прямоугольник");  for (int i = 0; i < 2; i++)  {  lPoint[i] = init.lPoint[i];  rPoint[i] = init.rPoint[i];  }  fPerimeter();  fSquare();  side = init.side;  }  Rectangles::~Rectangles()  {  delete[] name;  }  int Rectangles::getLXY(int index)  {  if (index >= 0 && index <= 1)  {  return lPoint[index];  }  return 0;  }  int Rectangles::getRXY(int index)  {  if (index >= 0 && index <= 1)  {  return rPoint[index];  }  return 0;  }  double Rectangles::getPerimeter()  {  return perim;  }  double Rectangles::getSquare()  {  return square;  }  void Rectangles::fPerimeter()  {  perim = 2 \* ((rPoint[0] - lPoint[0]) + (rPoint[1] - lPoint[1]));  }  void Rectangles::fSquare()  {  square = (rPoint[0] - lPoint[0]) \* (rPoint[1] - lPoint[1]);  }  void Rectangles::fMove(int x, int y)  {  lPoint[0] += x;  lPoint[1] += y;  rPoint[0] += x;  rPoint[1] += y;  }  void Rectangles::fAlteration(int x, int y)  {  rPoint[0] += x;  rPoint[1] += y;  fPerimeter();  fSquare();  } |

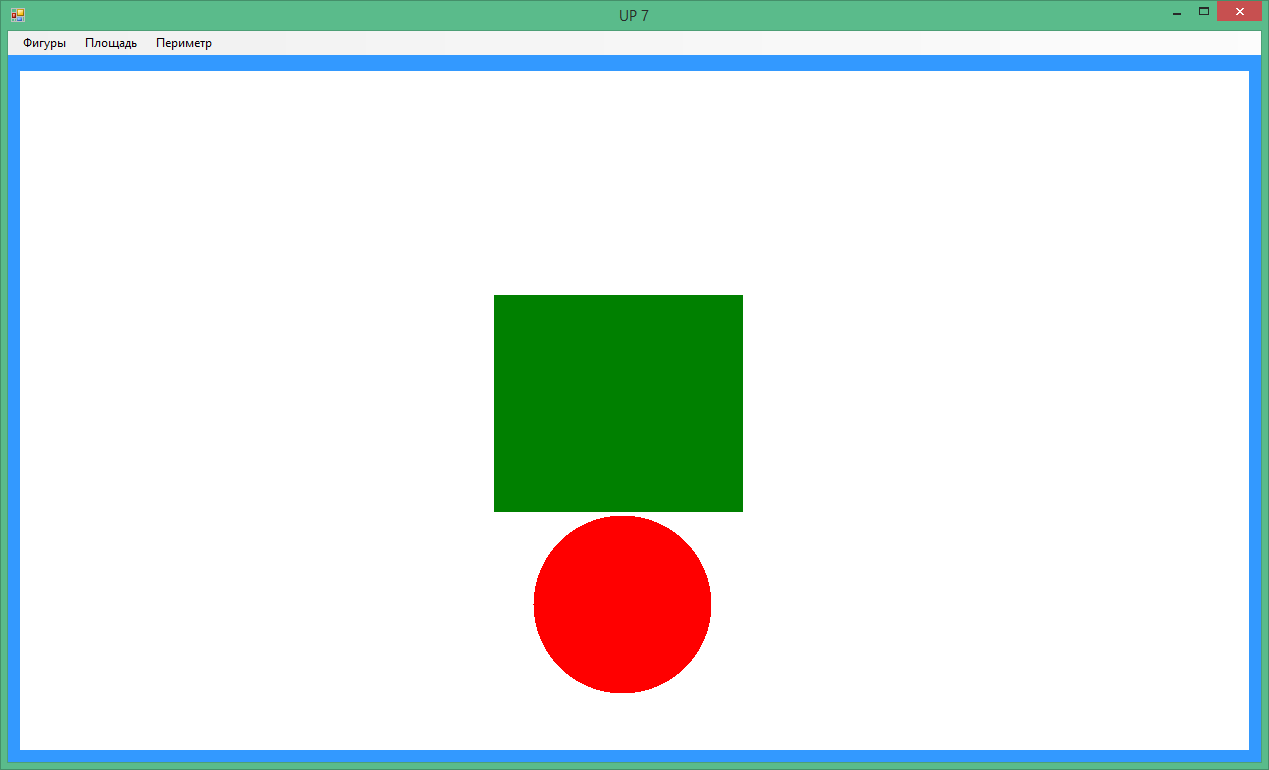
|  |
| --- |
| **Файл MyForm.cpp**  #include "MyForm.h"  using namespace System;  using namespace System::Windows::Forms;  [STAThreadAttribute]  int main(array<System::String ^> ^args)  {  Application::EnableVisualStyles();  Application::SetCompatibleTextRenderingDefault(false);  Sem\_Lab6WF::MyForm MyForm;  Application::Run(%MyForm);  return 0;  } |

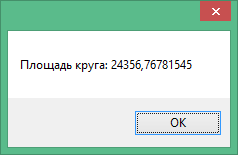
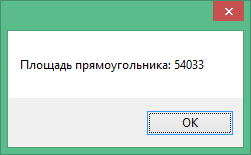
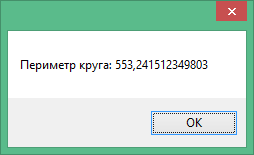
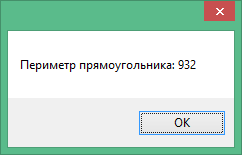
|  |
| --- |
| **Файл MyForm.h**  #include "Circle.h"  #include "Rectangle.h"  #pragma once  Circle circ(620, 340, 620, 380);  Rectangles rectan(580, 300, 660, 380);  bool cflag = false, rflag = false, cActive = false, rActive = false;  using namespace System;  using namespace System::ComponentModel;  using namespace System::Collections;  using namespace System::Windows::Forms;  using namespace System::Data;  using namespace System::Drawing;  namespace Sem\_Lab6WF {  /// <summary>  /// Summary for MyForm  /// </summary>  public ref class MyForm : public System::Windows::Forms::Form  {  Graphics^ gr;  public:  MyForm(void)  {  InitializeComponent();  gr = this->pictureBox1->CreateGraphics();  //  //TODO: Add the constructor code here  //  }  protected:  /// <summary>  /// Clean up any resources being used.  /// </summary>  ~MyForm()  {  if (components)  {  delete components;  }  }  private: System::Windows::Forms::PictureBox^ pictureBox1;  private: System::Windows::Forms::MenuStrip^ menuStrip1;  private: System::Windows::Forms::ToolStripMenuItem^ фигурыToolStripMenuItem;  private: System::Windows::Forms::ToolStripMenuItem^ кругToolStripMenuItem;  private: System::Windows::Forms::ToolStripMenuItem^ прямоугольникToolStripMenuItem;  private: System::Windows::Forms::ToolStripMenuItem^ очиститьToolStripMenuItem;  private: System::Windows::Forms::ToolStripMenuItem^ площадьToolStripMenuItem;  private: System::Windows::Forms::ToolStripMenuItem^ кругToolStripMenuItem1;  private: System::Windows::Forms::ToolStripMenuItem^ прямоугольникToolStripMenuItem1;  private: System::Windows::Forms::ToolStripMenuItem^ периметрToolStripMenuItem;  private: System::Windows::Forms::ToolStripMenuItem^ кругToolStripMenuItem2;  private: System::Windows::Forms::ToolStripMenuItem^ прямоугольникToolStripMenuItem2;  protected:  protected:  private:  /// <summary>  /// Required designer variable.  /// </summary>  System::ComponentModel::Container ^components;  #pragma region Windows Form Designer generated code  /// <summary>  /// Required method for Designer support - do not modify  /// the contents of this method with the code editor.  /// </summary>  void InitializeComponent(void)  {  this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());  this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());  this->фигурыToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());  this->кругToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());  this->прямоугольникToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());  this->очиститьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());  this->площадьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());  this->кругToolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripMenuItem());  this->прямоугольникToolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripMenuItem());  this->периметрToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());  this->кругToolStripMenuItem2 = (gcnew System::Windows::Forms::ToolStripMenuItem());  this->прямоугольникToolStripMenuItem2 = (gcnew System::Windows::Forms::ToolStripMenuItem());  (cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();  this->menuStrip1->SuspendLayout();  this->SuspendLayout();  //  // pictureBox1  //  this->pictureBox1->BackColor = System::Drawing::SystemColors::ControlLightLight;  this->pictureBox1->Location = System::Drawing::Point(12, 40);  this->pictureBox1->Name = L"pictureBox1";  this->pictureBox1->Size = System::Drawing::Size(1229, 679);  this->pictureBox1->TabIndex = 0;  this->pictureBox1->TabStop = false;  this->pictureBox1->MouseDown += gcnew MouseEventHandler(this, &MyForm::pictureBox1\_MouseDown);  this->pictureBox1->MouseUp += gcnew MouseEventHandler(this, &MyForm::pictureBox1\_MouseUp);  this->pictureBox1->MouseMove += gcnew MouseEventHandler(this, &MyForm::pictureBox1\_MouseMove);  this->pictureBox1->Paint += gcnew PaintEventHandler(this, &MyForm::pictureBox1\_Paint);  //  // menuStrip1  //  this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {  this->фигурыToolStripMenuItem,  this->площадьToolStripMenuItem, this->периметрToolStripMenuItem  });  this->menuStrip1->Location = System::Drawing::Point(0, 0);  this->menuStrip1->Name = L"menuStrip1";  this->menuStrip1->Size = System::Drawing::Size(1253, 24);  this->menuStrip1->TabIndex = 2;  this->menuStrip1->Text = L"menuStrip1";  //  // фигурыToolStripMenuItem  //  this->фигурыToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {  this->кругToolStripMenuItem,  this->прямоугольникToolStripMenuItem, this->очиститьToolStripMenuItem  });  this->фигурыToolStripMenuItem->Name = L"фигурыToolStripMenuItem";  this->фигурыToolStripMenuItem->Size = System::Drawing::Size(62, 20);  this->фигурыToolStripMenuItem->Text = L"Фигуры";  //  // кругToolStripMenuItem  //  this->кругToolStripMenuItem->Name = L"кругToolStripMenuItem";  this->кругToolStripMenuItem->Size = System::Drawing::Size(163, 22);  this->кругToolStripMenuItem->Text = L"Круг";  this->кругToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::кругToolStripMenuItem\_Click);  //  // прямоугольникToolStripMenuItem  //  this->прямоугольникToolStripMenuItem->Name = L"прямоугольникToolStripMenuItem";  this->прямоугольникToolStripMenuItem->Size = System::Drawing::Size(163, 22);  this->прямоугольникToolStripMenuItem->Text = L"Прямоугольник";  this->прямоугольникToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::прямоугольникToolStripMenuItem\_Click);  //  // очиститьToolStripMenuItem  //  this->очиститьToolStripMenuItem->Name = L"очиститьToolStripMenuItem";  this->очиститьToolStripMenuItem->Size = System::Drawing::Size(163, 22);  this->очиститьToolStripMenuItem->Text = L"Очистить";  this->очиститьToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::очиститьToolStripMenuItem\_Click);  //  // площадьToolStripMenuItem  //  this->площадьToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {  this->кругToolStripMenuItem1,  this->прямоугольникToolStripMenuItem1  });  this->площадьToolStripMenuItem->Name = L"площадьToolStripMenuItem";  this->площадьToolStripMenuItem->Size = System::Drawing::Size(71, 20);  this->площадьToolStripMenuItem->Text = L"Площадь";  //  // кругToolStripMenuItem1  //  this->кругToolStripMenuItem1->Name = L"кругToolStripMenuItem1";  this->кругToolStripMenuItem1->Size = System::Drawing::Size(163, 22);  this->кругToolStripMenuItem1->Text = L"Круг";  this->кругToolStripMenuItem1->Click += gcnew System::EventHandler(this, &MyForm::кругToolStripMenuItem1\_Click);  //  // прямоугольникToolStripMenuItem1  //  this->прямоугольникToolStripMenuItem1->Name = L"прямоугольникToolStripMenuItem1";  this->прямоугольникToolStripMenuItem1->Size = System::Drawing::Size(163, 22);  this->прямоугольникToolStripMenuItem1->Text = L"Прямоугольник";  this->прямоугольникToolStripMenuItem1->Click += gcnew System::EventHandler(this, &MyForm::прямоугольникToolStripMenuItem1\_Click);  //  // периметрToolStripMenuItem  //  this->периметрToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {  this->кругToolStripMenuItem2,  this->прямоугольникToolStripMenuItem2  });  this->периметрToolStripMenuItem->Name = L"периметрToolStripMenuItem";  this->периметрToolStripMenuItem->Size = System::Drawing::Size(75, 20);  this->периметрToolStripMenuItem->Text = L"Периметр";  //  // кругToolStripMenuItem2  //  this->кругToolStripMenuItem2->Name = L"кругToolStripMenuItem2";  this->кругToolStripMenuItem2->Size = System::Drawing::Size(163, 22);  this->кругToolStripMenuItem2->Text = L"Круг";  this->кругToolStripMenuItem2->Click += gcnew System::EventHandler(this, &MyForm::кругToolStripMenuItem2\_Click);  //  // прямоугольникToolStripMenuItem2  //  this->прямоугольникToolStripMenuItem2->Name = L"прямоугольникToolStripMenuItem2";  this->прямоугольникToolStripMenuItem2->Size = System::Drawing::Size(163, 22);  this->прямоугольникToolStripMenuItem2->Text = L"Прямоугольник";  this->прямоугольникToolStripMenuItem2->Click += gcnew System::EventHandler(this, &MyForm::прямоугольникToolStripMenuItem2\_Click);  //  // MyForm  //  this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);  this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;  this->AutoSizeMode = System::Windows::Forms::AutoSizeMode::GrowAndShrink;  this->BackColor = System::Drawing::SystemColors::MenuHighlight;  this->ClientSize = System::Drawing::Size(1253, 731);  this->Controls->Add(this->pictureBox1);  this->Controls->Add(this->menuStrip1);  this->MainMenuStrip = this->menuStrip1;  this->Name = L"MyForm";  this->Text = L"UP 7";  this->Load += gcnew System::EventHandler(this, &MyForm::MyForm\_Load);  (cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();  this->menuStrip1->ResumeLayout(false);  this->menuStrip1->PerformLayout();  this->ResumeLayout(false);  this->PerformLayout();  }  #pragma endregion  private: Void MyForm\_Load(Object^ sender, EventArgs^ e) {    }  private: System::Void кругToolStripMenuItem\_Click(Object^ sender, EventArgs^ e) {  cflag = true;  pictureBox1->Refresh();  }  private: System::Void прямоугольникToolStripMenuItem\_Click(Object^ sender, EventArgs^ e) {  rflag = true;  pictureBox1->Refresh();  }  private: System::Void очиститьToolStripMenuItem\_Click(Object^ sender, EventArgs^ e) {  cflag = false;  rflag = false;  pictureBox1->Refresh();  }  private: System::Void кругToolStripMenuItem1\_Click(Object^ sender, EventArgs^ e) {  if (!cflag)  {  MessageBox::Show("Круг не найден!");  }  else  {  MessageBox::Show("Площадь круга: " + Convert::ToString(circ.getSquare()));  }  }  private: System::Void прямоугольникToolStripMenuItem1\_Click(Object^ sender, EventArgs^ e) {  if (!rflag)  {  MessageBox::Show("Прямоугольник не найден!");  }  else  {  MessageBox::Show("Площадь прямоугольника: " + Convert::ToString(rectan.getSquare()));  }  }  private: System::Void кругToolStripMenuItem2\_Click(Object^ sender, EventArgs^ e) {  if (!cflag)  {  MessageBox::Show("Круг не найден!");  }  else  {  MessageBox::Show("Периметр круга: " + Convert::ToString(circ.getPerimeter()));  }  }  private: System::Void прямоугольникToolStripMenuItem2\_Click(Object^ sender, EventArgs^ e) {  if (!rflag)  {  MessageBox::Show("Прямоугольник не найден!");  }  else  {  MessageBox::Show("Периметр прямоугольника: " + Convert::ToString(rectan.getPerimeter()));  }  }  private: System::Void pictureBox1\_Paint(Object^ sender, PaintEventArgs^ e) {  if (cflag)  {  e->Graphics->FillEllipse(Brushes::Red,  circ.getLXY(0) - (int)circ.getR(), circ.getLXY(1) - (int)circ.getR(), 2 \* (int)circ.getR(), 2 \* (int)circ.getR());  }  if (rflag)  {  e->Graphics->FillRectangle(Brushes::Green,  rectan.getLXY(0), rectan.getLXY(1), rectan.getRXY(0) - rectan.getLXY(0), rectan.getRXY(1) - rectan.getLXY(1));  }  }  private: System::Void pictureBox1\_MouseDown(Object^ sender, MouseEventArgs^ e) {  if (e->Button == ::MouseButtons::Left && cflag  && abs(e->X - circ.getLXY(0)) <= circ.getR() && abs(e->Y - circ.getLXY(1)) <= circ.getR())  {  cActive = true;  }  if (e->Button == ::MouseButtons::Left && rflag  && e->X >= rectan.getLXY(0) && e->Y >= rectan.getLXY(1) && e->X <= rectan.getRXY(0) && e->Y <= rectan.getRXY(1))  {  rActive = true;  }  }  private: System::Void pictureBox1\_MouseUp(Object^ sender, MouseEventArgs^ e) {  cActive = false;  rActive = false;  }  private: System::Void pictureBox1\_MouseMove(Object^ sender, MouseEventArgs^ e) {  if (Control::ModifierKeys == Keys::Shift && cflag && (abs(e->X - circ.getRXY(0)) < 20 && abs(e->Y - circ.getRXY(1)) < 20))  {  ::Cursor::Current = Cursors::SizeNS;  }  if (Control::ModifierKeys == Keys::Shift && rflag && (abs(e->X - rectan.getRXY(0)) < 20 && abs(e->Y - rectan.getRXY(1)) < 20))  {  ::Cursor::Current = Cursors::SizeNWSE;  }  if (e->Button == ::MouseButtons::Left)  {  if (Control::ModifierKeys == Keys::Shift)  {  if (cActive && abs(e->X - circ.getRXY(0)) < 20 && abs(e->Y - circ.getRXY(1)) < 20)  {  circ.fAlteration(e->X - circ.getRXY(0), e->Y - circ.getRXY(1));  pictureBox1->Refresh();  }  if (rActive && abs(e->X - rectan.getRXY(0)) < 20 && abs(e->Y - rectan.getRXY(1)) < 20)  {  rectan.fAlteration(e->X - rectan.getRXY(0), e->Y - rectan.getRXY(1));  pictureBox1->Refresh();  }  }  else  {  if (cActive)  {  circ.fMove(e->X - circ.getLXY(0), e->Y - circ.getLXY(1));  pictureBox1->Refresh();  }  if (rActive)  {  rectan.fMove(e->X - rectan.getLXY(0), e->Y - rectan.getLXY(1));  pictureBox1->Refresh();  }  }  }  }  };  } |

## Скриншоты:







# Создание приложения MFC

Объявить абстрактный базовый класс «Фигура на плоскости», моделирующий общие свойства и поведение геометрических фигур. Включить в объявления классов следующее поля данных:

– название фигуры (отрезок, квадрат, круг и т.д.);

­– одно или несколько полей типа *Point*, характеризующие положение фигуры на плоскости. Например: центр круга (одно поле данных) или два поля для прямоугольника и квадрата (верхний левый и правый нижний углы фигуры);

– поля, характеризующие параметры фигуры: количество сторон многоугольника, периметр и площадь фигуры.

В класс наследник включить следующие методы:

– конструктор по умолчанию, конструктор с формальными параметрами и конструктор копирования, а также виртуальный деструктор;

– виртуальные методы для вычисления параметров объекта (периметра и площади фигуры с учетом ее вида; площади поверхностей и объем тела);

– виртуальный метод для отображения объекта на экране. В соответствующей функции-члене класса достаточно вывести текст «Рисунок <name> <координаты> », где name - название фигуры.

– виртуальный метод для моделирования перемещения фигуры на плоскости (сдвиг центра окружности или вершин четырехугольника); В соответствующей функции-члене класса достаточно выводить текст «Движение <name> <координаты> », где name - название фигуры.

– виртуальный метод для моделирования сжатия или растяжения фигуры (увеличение радиуса круга, уменьшение стороны квадрата).

## Код:

|  |
| --- |
| **Файл Figure.h**  #ifndef FIGURE\_H  #define FIGURE\_H  #define PI 3.14159265  #include <iostream>  #include <Windows.h>  #include <cmath>  using namespace std;  class Figure  {  protected:  char\* name;  int lPoint[2];  int rPoint[2];  int side;  double perim;  double square;  public:  virtual void fPerimeter() = 0;  virtual void fSquare() = 0;  virtual void fMove(int x, int y) = 0;  virtual void fAlteration(int x, int y) = 0;  };  #endif |

|  |  |
| --- | --- |
| **Файл Circle.h**  #ifndef CIRCLE\_H  #define CIRCLE\_H  #include "Figure.h"  class Circle : public Figure  {  double R;  public:  Circle();  Circle(int lx, int ly, int rx, int ry);  Circle(const Circle& init);  virtual ~Circle();  void cRadius();  int getLXY(int index);  int getRXY(int index);  double getPerimeter();  double getSquare();  double getR();  virtual void fPerimeter();  virtual void fSquare();  virtual void fMove(int x, int y);  virtual void fAlteration(int x, int y);  };  #endif | **Файл Circle.cpp**  #include "stdafx.h"  #include "Circle.h"  Circle::Circle()  {  name = new char[strlen("Круг") + 1];  strcpy(name, "Круг");  for (int i = 0; i < 2; i++)  {  lPoint[i] = 0;  rPoint[i] = 0;  }  side = 1;  }  Circle::Circle(int lx, int ly, int rx, int ry)  {  name = new char[strlen("Круг") + 1];  strcpy(name, "Круг");  lPoint[0] = lx;  lPoint[1] = ly;  rPoint[0] = rx;  rPoint[1] = ry;  cRadius();  fPerimeter();  fSquare();  side = 1;  }  Circle::Circle(const Circle& init)  {  name = new char[strlen("Круг") + 1];  strcpy(name, "Круг");  for (int i = 0; i < 2; i++)  {  lPoint[i] = init.lPoint[i];  rPoint[i] = init.rPoint[i];  }  R = init.R;  side = 1;  }  Circle::~Circle()  {  delete[] name;  }  void Circle::cRadius()  {  R = sqrt((rPoint[0] - lPoint[0])\*(rPoint[0] - lPoint[0]) + (rPoint[1] - lPoint[1])\*(rPoint[1] - lPoint[1]));  }  int Circle::getLXY(int index)  {  if (index >= 0 && index <= 1)  {  return lPoint[index];  }  return 0;  }  int Circle::getRXY(int index)  {  if (index >= 0 && index <= 1)  {  return rPoint[index];  }  return 0;  }  double Circle::getPerimeter()  {  return perim;  }  double Circle::getSquare()  {  return square;  }  double Circle::getR()  {  return R;  }  void Circle::fPerimeter()  {  perim = 2 \* PI \* R;  }  void Circle::fSquare()  {  square = PI \* R \* R;  }  void Circle::fMove(int x, int y)  {  lPoint[0] += x;  lPoint[1] += y;  rPoint[0] += x;  rPoint[1] += y;  }  void Circle::fAlteration(int x, int y)  {  rPoint[0] += x;  rPoint[1] += y;  fPerimeter();  fSquare();  cRadius();  } |

|  |  |
| --- | --- |
| **Файл Rectangle.h**  #ifndef RECTANGLE\_H  #define RECTANGLE\_H  #include "Figure.h"  class Rectangles : public Figure  {  public:  Rectangles();  Rectangles(int lx, int ly, int rx, int ry);  Rectangles(const Rectangles& init);  virtual ~Rectangles();  int getLXY(int index);  int getRXY(int index);  double getPerimeter();  double getSquare();  virtual void fPerimeter();  virtual void fSquare();  virtual void fMove(int x, int y);  virtual void fAlteration(int x, int y);  };  #endif | **Файл Rectangle.cpp**  #include "stdafx.h"  #include "Rectangle.h"  Rectangles::Rectangles()  {  name = new char[strlen("Прямоугольник") + 1];  strcpy(name, "Прямоугольник");  for (int i = 0; i < 2; i++)  {  lPoint[i] = 0;  rPoint[i] = 0;  }  side = 4;  }  Rectangles::Rectangles(int lx, int ly, int rx, int ry)  {  name = new char[strlen("Прямоугольник") + 1];  strcpy(name, "Прямоугольник");  lPoint[0] = lx;  lPoint[1] = ly;  rPoint[0] = rx;  rPoint[1] = ry;  fPerimeter();  fSquare();  side = 4;  }  Rectangles::Rectangles(const Rectangles& init)  {  name = new char[strlen("Прямоугольник") + 1];  strcpy(name, "Прямоугольник");  for (int i = 0; i < 2; i++)  {  lPoint[i] = init.lPoint[i];  rPoint[i] = init.rPoint[i];  }  fPerimeter();  fSquare();  side = init.side;  }  Rectangles::~Rectangles()  {  delete[] name;  }  int Rectangles::getLXY(int index)  {  if (index >= 0 && index <= 1)  {  return lPoint[index];  }  return 0;  }  int Rectangles::getRXY(int index)  {  if (index >= 0 && index <= 1)  {  return rPoint[index];  }  return 0;  }  double Rectangles::getPerimeter()  {  return perim;  }  double Rectangles::getSquare()  {  return square;  }  void Rectangles::fPerimeter()  {  perim = 2 \* ((rPoint[0] - lPoint[0]) + (rPoint[1] - lPoint[1]));  }  void Rectangles::fSquare()  {  square = (rPoint[0] - lPoint[0]) \* (rPoint[1] - lPoint[1]);  }  void Rectangles::fMove(int x, int y)  {  lPoint[0] += x;  lPoint[1] += y;  rPoint[0] += x;  rPoint[1] += y;  }  void Rectangles::fAlteration(int x, int y)  {  rPoint[0] += x;  rPoint[1] += y;  fPerimeter();  fSquare();  } |

|  |  |
| --- | --- |
| **Файл ChildFrm.h**  // ChildFrm.h : interface of the CChildFrame class  //  #pragma once  class CChildFrame : public CMDIChildWnd  {  DECLARE\_DYNCREATE(CChildFrame)  public:  CChildFrame();  // Attributes  public:  // Operations  public:  // Overrides  virtual BOOL PreCreateWindow(CREATESTRUCT& cs);  // Implementation  public:  virtual ~CChildFrame();  #ifdef \_DEBUG  virtual void AssertValid() const;  virtual void Dump(CDumpContext& dc) const;  #endif  // Generated message map functions  protected:  DECLARE\_MESSAGE\_MAP()  }; | **Файл ChildFrm.cpp**  // ChildFrm.cpp : implementation of the CChildFrame class  //  #include "stdafx.h"  #include "UP\_8.h"  #include "ChildFrm.h"  #ifdef \_DEBUG  #define new DEBUG\_NEW  #endif  // CChildFrame  IMPLEMENT\_DYNCREATE(CChildFrame, CMDIChildWnd)  BEGIN\_MESSAGE\_MAP(CChildFrame, CMDIChildWnd)  END\_MESSAGE\_MAP()  // CChildFrame construction/destruction  CChildFrame::CChildFrame()  {  // TODO: add member initialization code here  }  CChildFrame::~CChildFrame()  {  }  BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)  {  // TODO: Modify the Window class or styles here by modifying the CREATESTRUCT cs  if( !CMDIChildWnd::PreCreateWindow(cs) )  return FALSE;  return TRUE;  }  // CChildFrame diagnostics  #ifdef \_DEBUG  void CChildFrame::AssertValid() const  {  CMDIChildWnd::AssertValid();  }  void CChildFrame::Dump(CDumpContext& dc) const  {  CMDIChildWnd::Dump(dc);  }  #endif //\_DEBUG  // CChildFrame message handlers |

|  |  |
| --- | --- |
| **Файл MainFrm.h**  // MainFrm.h : interface of the CMainFrame class  //  #pragma once  class CMainFrame : public CMDIFrameWnd  {  DECLARE\_DYNAMIC(CMainFrame)  public:  CMainFrame();  // Attributes  public:  // Operations  public:  // Overrides  public:  virtual BOOL PreCreateWindow(CREATESTRUCT& cs);  // Implementation  public:  virtual ~CMainFrame();  #ifdef \_DEBUG  virtual void AssertValid() const;  virtual void Dump(CDumpContext& dc) const;  #endif  protected: // control bar embedded members  CToolBar m\_wndToolBar;  CStatusBar m\_wndStatusBar;  // Generated message map functions  protected:  afx\_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);  afx\_msg void OnApplicationLook(UINT id);  afx\_msg void OnUpdateApplicationLook(CCmdUI\* pCmdUI);  DECLARE\_MESSAGE\_MAP()  }; | **Файл MainFrm.cpp**  // MainFrm.cpp : implementation of the CMainFrame class  //  #include "stdafx.h"  #include "UP\_8.h"  #include "MainFrm.h"  #ifdef \_DEBUG  #define new DEBUG\_NEW  #endif  // CMainFrame  IMPLEMENT\_DYNAMIC(CMainFrame, CMDIFrameWnd)  const int iMaxUserToolbars = 10;  const UINT uiFirstUserToolBarId = AFX\_IDW\_CONTROLBAR\_FIRST + 40;  const UINT uiLastUserToolBarId = uiFirstUserToolBarId + iMaxUserToolbars - 1;  BEGIN\_MESSAGE\_MAP(CMainFrame, CMDIFrameWnd)  ON\_WM\_CREATE()  ON\_COMMAND\_RANGE(ID\_VIEW\_APPLOOK\_WIN\_2000, ID\_VIEW\_APPLOOK\_WINDOWS\_7, &CMainFrame::OnApplicationLook)  ON\_UPDATE\_COMMAND\_UI\_RANGE(ID\_VIEW\_APPLOOK\_WIN\_2000, ID\_VIEW\_APPLOOK\_WINDOWS\_7, &CMainFrame::OnUpdateApplicationLook)  END\_MESSAGE\_MAP()  static UINT indicators[] =  {  ID\_SEPARATOR, // status line indicator  ID\_INDICATOR\_CAPS,  ID\_INDICATOR\_NUM,  ID\_INDICATOR\_SCRL,  };  // CMainFrame construction/destruction  CMainFrame::CMainFrame()  {  // TODO: add member initialization code here  }  CMainFrame::~CMainFrame()  {  }  int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)  {  if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)  return -1;  if (!m\_wndToolBar.CreateEx(this, TBSTYLE\_FLAT, WS\_CHILD | WS\_VISIBLE | CBRS\_TOP | CBRS\_GRIPPER | CBRS\_TOOLTIPS | CBRS\_FLYBY | CBRS\_SIZE\_DYNAMIC) ||  !m\_wndToolBar.LoadToolBar(IDR\_MAINFRAME))  {  TRACE0("Failed to create toolbar\n");  return -1; // fail to create  }  if (!m\_wndStatusBar.Create(this))  {  TRACE0("Failed to create status bar\n");  return -1; // fail to create  }  m\_wndStatusBar.SetIndicators(indicators, sizeof(indicators)/sizeof(UINT));  // TODO: Delete these three lines if you don't want the toolbar to be dockable  m\_wndToolBar.EnableDocking(CBRS\_ALIGN\_ANY);  EnableDocking(CBRS\_ALIGN\_ANY);  DockControlBar(&m\_wndToolBar);  return 0;  }  BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)  {  if( !CMDIFrameWnd::PreCreateWindow(cs) )  return FALSE;  // TODO: Modify the Window class or styles here by modifying  // the CREATESTRUCT cs  return TRUE;  }  // CMainFrame diagnostics  #ifdef \_DEBUG  void CMainFrame::AssertValid() const  {  CMDIFrameWnd::AssertValid();  }  void CMainFrame::Dump(CDumpContext& dc) const  {  CMDIFrameWnd::Dump(dc);  }  #endif //\_DEBUG  // CMainFrame message handlers  void CMainFrame::OnApplicationLook(UINT id)  {  CWaitCursor wait;  theApp.m\_nAppLook = id;  switch (theApp.m\_nAppLook)  {  case ID\_VIEW\_APPLOOK\_WIN\_2000:  CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManager));  break;  case ID\_VIEW\_APPLOOK\_OFF\_XP:  CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerOfficeXP));  break;  case ID\_VIEW\_APPLOOK\_WIN\_XP:  CMFCVisualManagerWindows::m\_b3DTabsXPTheme = TRUE;  CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerWindows));  break;  case ID\_VIEW\_APPLOOK\_OFF\_2003:  CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerOffice2003));  CDockingManager::SetDockingMode(DT\_SMART);  break;  case ID\_VIEW\_APPLOOK\_VS\_2005:  CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerVS2005));  CDockingManager::SetDockingMode(DT\_SMART);  break;  case ID\_VIEW\_APPLOOK\_VS\_2008:  CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerVS2008));  CDockingManager::SetDockingMode(DT\_SMART);  break;  case ID\_VIEW\_APPLOOK\_WINDOWS\_7:  CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerWindows7));  CDockingManager::SetDockingMode(DT\_SMART);  break;  default:  switch (theApp.m\_nAppLook)  {  case ID\_VIEW\_APPLOOK\_OFF\_2007\_BLUE:  CMFCVisualManagerOffice2007::SetStyle(CMFCVisualManagerOffice2007::Office2007\_LunaBlue);  break;  case ID\_VIEW\_APPLOOK\_OFF\_2007\_BLACK:  CMFCVisualManagerOffice2007::SetStyle(CMFCVisualManagerOffice2007::Office2007\_ObsidianBlack);  break;  case ID\_VIEW\_APPLOOK\_OFF\_2007\_SILVER:  CMFCVisualManagerOffice2007::SetStyle(CMFCVisualManagerOffice2007::Office2007\_Silver);  break;  case ID\_VIEW\_APPLOOK\_OFF\_2007\_AQUA:  CMFCVisualManagerOffice2007::SetStyle(CMFCVisualManagerOffice2007::Office2007\_Aqua);  break;  }  CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerOffice2007));  CDockingManager::SetDockingMode(DT\_SMART);  }  RedrawWindow(NULL, NULL, RDW\_ALLCHILDREN | RDW\_INVALIDATE | RDW\_UPDATENOW | RDW\_FRAME | RDW\_ERASE);  }  void CMainFrame::OnUpdateApplicationLook(CCmdUI\* pCmdUI)  {  pCmdUI->SetRadio(theApp.m\_nAppLook == pCmdUI->m\_nID);  } |

|  |  |
| --- | --- |
| **Файл stdafx.h**  // stdafx.h : include file for standard system include files,  // or project specific include files that are used frequently,  // but are changed infrequently  #pragma once  #ifndef VC\_EXTRALEAN  #define VC\_EXTRALEAN // Exclude rarely-used stuff from Windows headers  #endif  #include "targetver.h"  #define \_ATL\_CSTRING\_EXPLICIT\_CONSTRUCTORS // some CString constructors will be explicit  // turns off MFC's hiding of some common and often safely ignored warning messages  #define \_AFX\_ALL\_WARNINGS  #include <afxwin.h> // MFC core and standard components  #include <afxext.h> // MFC extensions  #include <afxdisp.h> // MFC Automation classes  #ifndef \_AFX\_NO\_OLE\_SUPPORT  #include <afxdtctl.h> // MFC support for Internet Explorer 4 Common Controls  #endif  #ifndef \_AFX\_NO\_AFXCMN\_SUPPORT  #include <afxcmn.h> // MFC support for Windows Common Controls  #endif // \_AFX\_NO\_AFXCMN\_SUPPORT  #include <afxcontrolbars.h> // MFC support for ribbons and control bars  #ifdef \_UNICODE  #if defined \_M\_IX86  #pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='x86' publicKeyToken='6595b64144ccf1df' language='\*'\"")  #elif defined \_M\_X64  #pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='amd64' publicKeyToken='6595b64144ccf1df' language='\*'\"")  #else  #pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='\*' publicKeyToken='6595b64144ccf1df' language='\*'\"")  #endif  #endif | **Файл stdafx.cpp**  // stdafx.cpp : source file that includes just the standard includes  // UP\_8.pch will be the pre-compiled header  // stdafx.obj will contain the pre-compiled type information  #include "stdafx.h" |

|  |
| --- |
| **Файл targertver.h**  #pragma once  // Including SDKDDKVer.h defines the highest available Windows platform.  // If you wish to build your application for a previous Windows platform, include WinSDKVer.h and  // set the \_WIN32\_WINNT macro to the platform you wish to support before including SDKDDKVer.h.  #include <SDKDDKVer.h> |

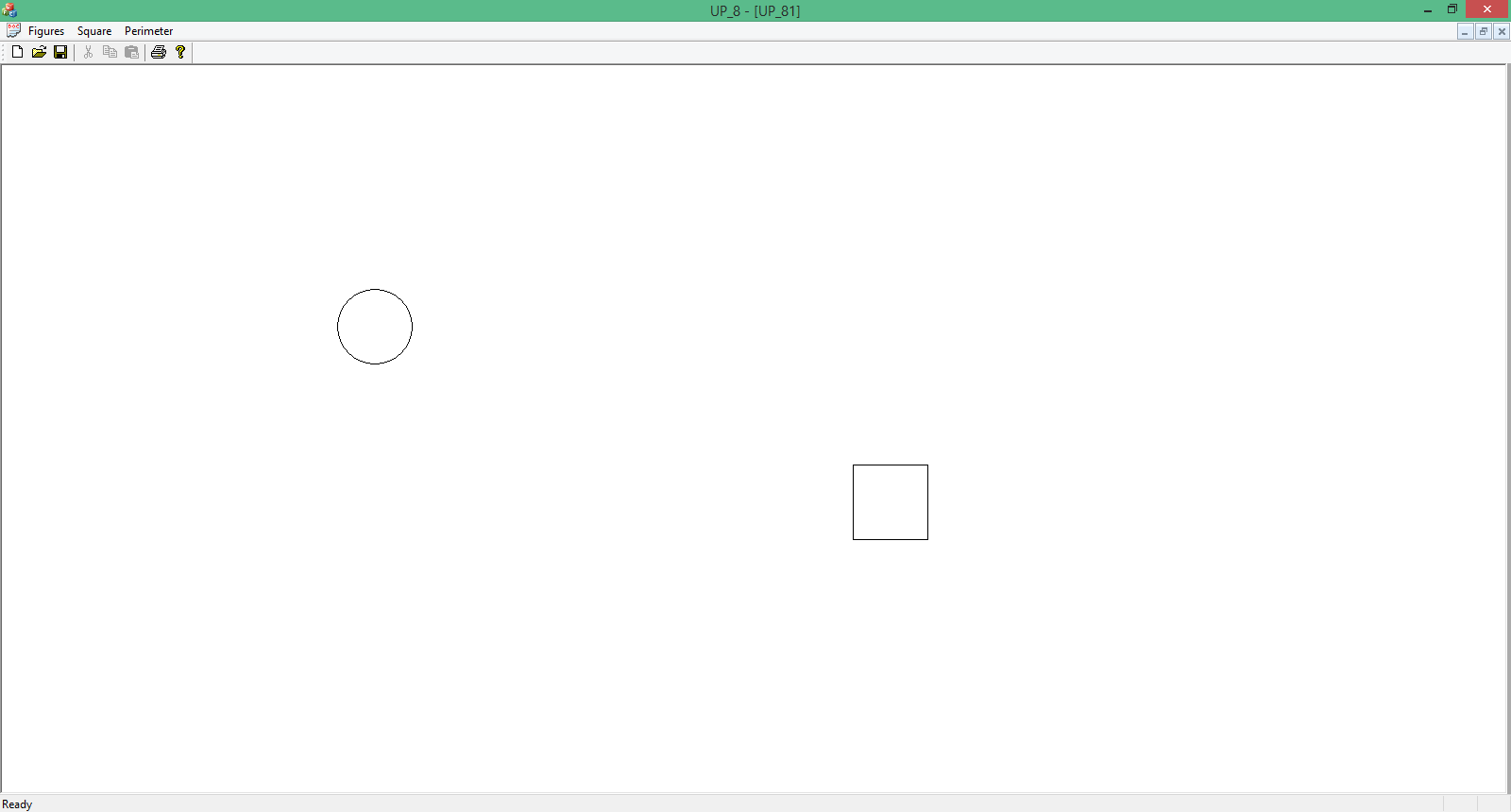
|  |
| --- |
| **Файл Resours.h**  //{{NO\_DEPENDENCIES}}  // Microsoft Visual C++ generated include file.  // Used by UP\_8.rc  //  #define IDD\_ABOUTBOX 100  #define IDP\_OLE\_INIT\_FAILED 100  #define IDR\_MAINFRAME 128  #define IDR\_UP\_8TYPE 130  #define ID\_WINDOW\_MANAGER 131  #define IDR\_THEME\_MENU 200  #define ID\_SET\_STYLE 201  #define ID\_VIEW\_APPLOOK\_WIN\_2000 205  #define ID\_VIEW\_APPLOOK\_OFF\_XP 206  #define ID\_VIEW\_APPLOOK\_WIN\_XP 207  #define ID\_VIEW\_APPLOOK\_OFF\_2003 208  #define ID\_VIEW\_APPLOOK\_VS\_2005 209  #define ID\_VIEW\_APPLOOK\_VS\_2008 210  #define ID\_VIEW\_APPLOOK\_OFF\_2007\_BLUE 215  #define ID\_VIEW\_APPLOOK\_OFF\_2007\_BLACK 216  #define ID\_VIEW\_APPLOOK\_OFF\_2007\_SILVER 217  #define ID\_VIEW\_APPLOOK\_OFF\_2007\_AQUA 218  #define ID\_VIEW\_APPLOOK\_WINDOWS\_7 219  #define ID\_32771 32771  #define ID\_32772 32772  #define ID\_32773 32773  #define ID\_32774 32774  #define ID\_32775 32775  #define ID\_32776 32776  #define ID\_32777 32777  #define ID\_32778 32778  #define ID\_32779 32779  #define ID\_32780 32780  #define ID\_32781 32781  #define ID\_FIGURES\_CIRCLE 32782  #define ID\_FIGURES\_RECTANGLE 32783  #define ID\_FIGURES\_CLEAN 32784  #define ID\_SQUARE\_CIRCLE 32785  #define ID\_SQUARE\_RECTANGLE 32786  #define ID\_PERIMETER\_CIRCLE 32787  #define ID\_PERIMETER\_RECTANGLE 32788  // Next default values for new objects  //  #ifdef APSTUDIO\_INVOKED  #ifndef APSTUDIO\_READONLY\_SYMBOLS  #define \_APS\_NEXT\_RESOURCE\_VALUE 310  #define \_APS\_NEXT\_COMMAND\_VALUE 32790  #define \_APS\_NEXT\_CONTROL\_VALUE 1000  #define \_APS\_NEXT\_SYMED\_VALUE 310  #endif  #endif |

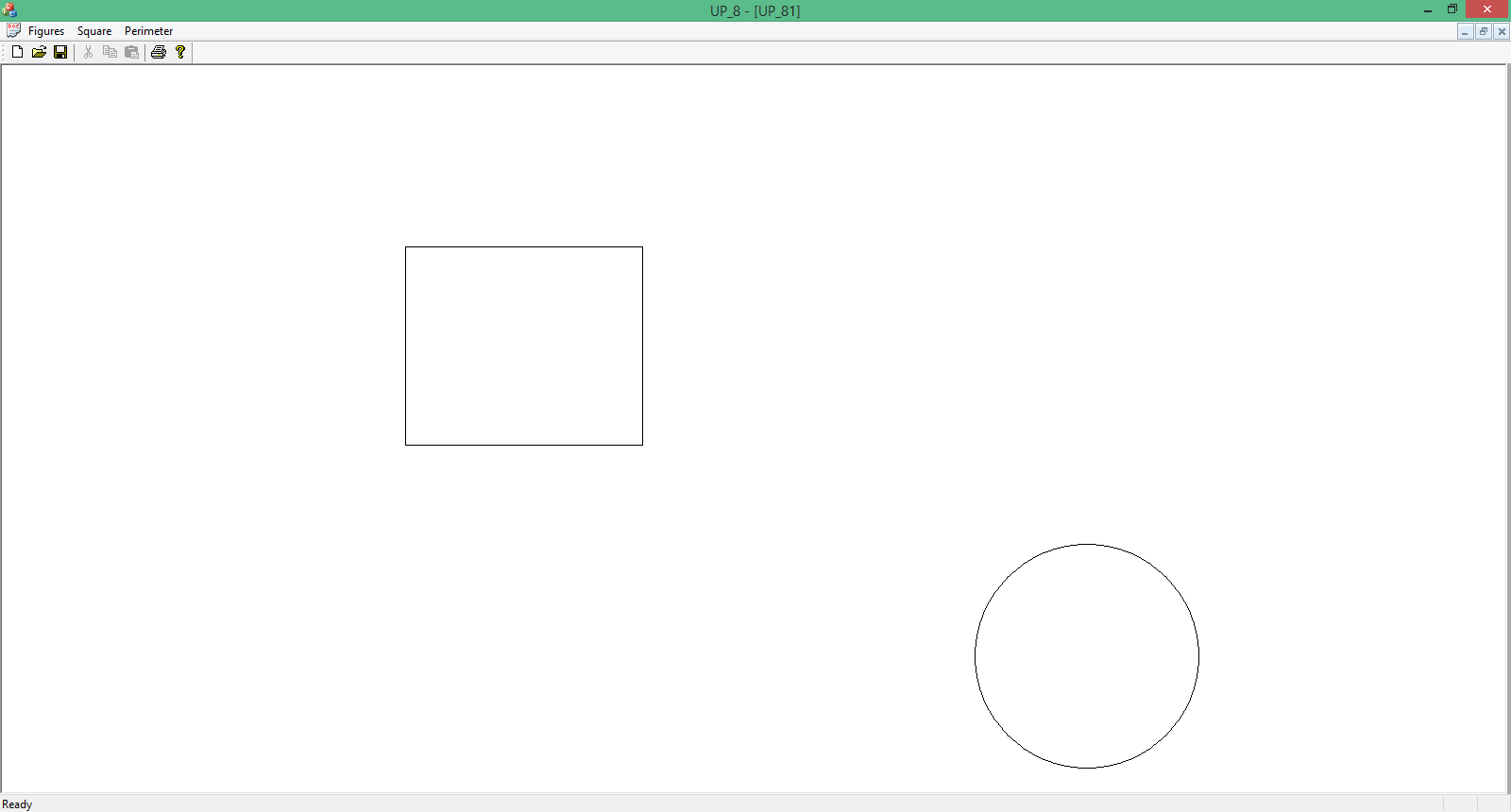
|  |  |
| --- | --- |
| **Файл UP\_8.h**  // UP\_8.h : main header file for the UP\_8 application  //  #pragma once  #ifndef \_\_AFXWIN\_H\_\_  #error "include 'stdafx.h' before including this file for PCH"  #endif  #include "resource.h" // main symbols  // CUP\_8App:  // See UP\_8.cpp for the implementation of this class  //  class CUP\_8App : public CWinApp  {  public:  CUP\_8App();  // Overrides  public:  virtual BOOL InitInstance();  virtual int ExitInstance();  // Implementation  UINT m\_nAppLook;  afx\_msg void OnAppAbout();  DECLARE\_MESSAGE\_MAP()  };  extern CUP\_8App theApp; | **Файл UP\_8.cpp**  // UP\_8.cpp : Defines the class behaviors for the application.  //  #include "stdafx.h"  #include "afxwinappex.h"  #include "afxdialogex.h"  #include "UP\_8.h"  #include "MainFrm.h"  #include "ChildFrm.h"  #include "UP\_8Doc.h"  #include "UP\_8View.h"  #ifdef \_DEBUG  #define new DEBUG\_NEW  #endif  // CUP\_8App  BEGIN\_MESSAGE\_MAP(CUP\_8App, CWinApp)  ON\_COMMAND(ID\_APP\_ABOUT, &CUP\_8App::OnAppAbout)  // Standard file based document commands  ON\_COMMAND(ID\_FILE\_NEW, &CWinApp::OnFileNew)  ON\_COMMAND(ID\_FILE\_OPEN, &CWinApp::OnFileOpen)  // Standard print setup command  ON\_COMMAND(ID\_FILE\_PRINT\_SETUP, &CWinApp::OnFilePrintSetup)  END\_MESSAGE\_MAP()  // CUP\_8App construction  CUP\_8App::CUP\_8App()  {  // support Restart Manager  m\_dwRestartManagerSupportFlags = AFX\_RESTART\_MANAGER\_SUPPORT\_ALL\_ASPECTS;  #ifdef \_MANAGED  // If the application is built using Common Language Runtime support (/clr):  // 1) This additional setting is needed for Restart Manager support to work properly.  // 2) In your project, you must add a reference to System.Windows.Forms in order to build.  System::Windows::Forms::Application::SetUnhandledExceptionMode(System::Windows::Forms::UnhandledExceptionMode::ThrowException);  #endif  // TODO: replace application ID string below with unique ID string; recommended  // format for string is CompanyName.ProductName.SubProduct.VersionInformation  SetAppID(\_T("UP\_8.AppID.NoVersion"));  // TODO: add construction code here,  // Place all significant initialization in InitInstance  }  // The one and only CUP\_8App object  CUP\_8App theApp;  // CUP\_8App initialization  BOOL CUP\_8App::InitInstance()  {  // InitCommonControlsEx() is required on Windows XP if an application  // manifest specifies use of ComCtl32.dll version 6 or later to enable  // visual styles. Otherwise, any window creation will fail.  INITCOMMONCONTROLSEX InitCtrls;  InitCtrls.dwSize = sizeof(InitCtrls);  // Set this to include all the common control classes you want to use  // in your application.  InitCtrls.dwICC = ICC\_WIN95\_CLASSES;  InitCommonControlsEx(&InitCtrls);  CWinApp::InitInstance();  // Initialize OLE libraries  if (!AfxOleInit())  {  AfxMessageBox(IDP\_OLE\_INIT\_FAILED);  return FALSE;  }  AfxEnableControlContainer();  EnableTaskbarInteraction(FALSE);  // AfxInitRichEdit2() is required to use RichEdit control  // AfxInitRichEdit2();  // Standard initialization  // If you are not using these features and wish to reduce the size  // of your final executable, you should remove from the following  // the specific initialization routines you do not need  // Change the registry key under which our settings are stored  // TODO: You should modify this string to be something appropriate  // such as the name of your company or organization  SetRegistryKey(\_T("Local AppWizard-Generated Applications"));  LoadStdProfileSettings(4); // Load standard INI file options (including MRU)  // Register the application's document templates. Document templates  // serve as the connection between documents, frame windows and views  CMultiDocTemplate\* pDocTemplate;  pDocTemplate = new CMultiDocTemplate(IDR\_UP\_8TYPE,  RUNTIME\_CLASS(CUP\_8Doc),  RUNTIME\_CLASS(CChildFrame), // custom MDI child frame  RUNTIME\_CLASS(CUP\_8View));  if (!pDocTemplate)  return FALSE;  AddDocTemplate(pDocTemplate);  // create main MDI Frame window  CMainFrame\* pMainFrame = new CMainFrame;  if (!pMainFrame || !pMainFrame->LoadFrame(IDR\_MAINFRAME))  {  delete pMainFrame;  return FALSE;  }  m\_pMainWnd = pMainFrame;  // Parse command line for standard shell commands, DDE, file open  CCommandLineInfo cmdInfo;  ParseCommandLine(cmdInfo);  // Dispatch commands specified on the command line. Will return FALSE if  // app was launched with /RegServer, /Register, /Unregserver or /Unregister.  if (!ProcessShellCommand(cmdInfo))  return FALSE;  // The main window has been initialized, so show and update it  pMainFrame->ShowWindow(m\_nCmdShow);  pMainFrame->UpdateWindow();  return TRUE;  }  int CUP\_8App::ExitInstance()  {  //TODO: handle additional resources you may have added  AfxOleTerm(FALSE);  return CWinApp::ExitInstance();  }  // CUP\_8App message handlers  // CAboutDlg dialog used for App About  class CAboutDlg : public CDialogEx  {  public:  CAboutDlg();  // Dialog Data  enum { IDD = IDD\_ABOUTBOX };  protected:  virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV support  // Implementation  protected:  DECLARE\_MESSAGE\_MAP()  };  CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD)  {  }  void CAboutDlg::DoDataExchange(CDataExchange\* pDX)  {  CDialogEx::DoDataExchange(pDX);  }  BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)  END\_MESSAGE\_MAP()  // App command to run the dialog  void CUP\_8App::OnAppAbout()  {  CAboutDlg aboutDlg;  aboutDlg.DoModal();  }  // CUP\_8App message handlers |

|  |  |
| --- | --- |
| **Файл UP\_8Doc.h**  // UP\_8Doc.h : interface of the CUP\_8Doc class  //  #pragma once  class CUP\_8Doc : public CDocument  {  protected: // create from serialization only  CUP\_8Doc();  DECLARE\_DYNCREATE(CUP\_8Doc)  // Attributes  public:  // Operations  public:  // Overrides  public:  virtual BOOL OnNewDocument();  virtual void Serialize(CArchive& ar);  #ifdef SHARED\_HANDLERS  virtual void InitializeSearchContent();  virtual void OnDrawThumbnail(CDC& dc, LPRECT lprcBounds);  #endif // SHARED\_HANDLERS  // Implementation  public:  virtual ~CUP\_8Doc();  #ifdef \_DEBUG  virtual void AssertValid() const;  virtual void Dump(CDumpContext& dc) const;  #endif  protected:  // Generated message map functions  protected:  DECLARE\_MESSAGE\_MAP()  #ifdef SHARED\_HANDLERS  // Helper function that sets search content for a Search Handler  void SetSearchContent(const CString& value);  #endif // SHARED\_HANDLERS  }; | **Файл UP\_8Doc.cpp**  // UP\_8Doc.cpp : implementation of the CUP\_8Doc class  //  #include "stdafx.h"  // SHARED\_HANDLERS can be defined in an ATL project implementing preview, thumbnail  // and search filter handlers and allows sharing of document code with that project.  #ifndef SHARED\_HANDLERS  #include "UP\_8.h"  #endif  #include "UP\_8Doc.h"  #include <propkey.h>  #ifdef \_DEBUG  #define new DEBUG\_NEW  #endif  // CUP\_8Doc  IMPLEMENT\_DYNCREATE(CUP\_8Doc, CDocument)  BEGIN\_MESSAGE\_MAP(CUP\_8Doc, CDocument)  END\_MESSAGE\_MAP()  // CUP\_8Doc construction/destruction  CUP\_8Doc::CUP\_8Doc()  {  // TODO: add one-time construction code here  }  CUP\_8Doc::~CUP\_8Doc()  {  }  BOOL CUP\_8Doc::OnNewDocument()  {  if (!CDocument::OnNewDocument())  return FALSE;  // TODO: add reinitialization code here  // (SDI documents will reuse this document)  return TRUE;  }  // CUP\_8Doc serialization  void CUP\_8Doc::Serialize(CArchive& ar)  {  if (ar.IsStoring())  {  // TODO: add storing code here  }  else  {  // TODO: add loading code here  }  }  #ifdef SHARED\_HANDLERS  // Support for thumbnails  void CUP\_8Doc::OnDrawThumbnail(CDC& dc, LPRECT lprcBounds)  {  // Modify this code to draw the document's data  dc.FillSolidRect(lprcBounds, RGB(255, 255, 255));  CString strText = \_T("TODO: implement thumbnail drawing here");  LOGFONT lf;  CFont\* pDefaultGUIFont = CFont::FromHandle((HFONT) GetStockObject(DEFAULT\_GUI\_FONT));  pDefaultGUIFont->GetLogFont(&lf);  lf.lfHeight = 36;  CFont fontDraw;  fontDraw.CreateFontIndirect(&lf);  CFont\* pOldFont = dc.SelectObject(&fontDraw);  dc.DrawText(strText, lprcBounds, DT\_CENTER | DT\_WORDBREAK);  dc.SelectObject(pOldFont);  }  // Support for Search Handlers  void CUP\_8Doc::InitializeSearchContent()  {  CString strSearchContent;  // Set search contents from document's data.  // The content parts should be separated by ";"  // For example: strSearchContent = \_T("point;rectangle;circle;ole object;");  SetSearchContent(strSearchContent);  }  void CUP\_8Doc::SetSearchContent(const CString& value)  {  if (value.IsEmpty())  {  RemoveChunk(PKEY\_Search\_Contents.fmtid, PKEY\_Search\_Contents.pid);  }  else  {  CMFCFilterChunkValueImpl \*pChunk = NULL;  ATLTRY(pChunk = new CMFCFilterChunkValueImpl);  if (pChunk != NULL)  {  pChunk->SetTextValue(PKEY\_Search\_Contents, value, CHUNK\_TEXT);  SetChunkValue(pChunk);  }  }  }  #endif // SHARED\_HANDLERS  // CUP\_8Doc diagnostics  #ifdef \_DEBUG  void CUP\_8Doc::AssertValid() const  {  CDocument::AssertValid();  }  void CUP\_8Doc::Dump(CDumpContext& dc) const  {  CDocument::Dump(dc);  }  #endif //\_DEBUG  // CUP\_8Doc commands |

|  |  |
| --- | --- |
| **Файл UP\_8View.h**  // UP\_8View.h : interface of the CUP\_8View class  //  #pragma once  #include "Circle.h"  #include "Rectangle.h"  //Circle circ(620, 340, 620, 380);  //Rectangles rectan(580, 300, 660, 380);  class CUP\_8View : public CView  {  protected: // create from serialization only  CUP\_8View();  DECLARE\_DYNCREATE(CUP\_8View)  // Attributes  public:  CUP\_8Doc\* GetDocument() const;  // Operations  public:  // Overrides  public:  virtual void OnDraw(CDC\* pDC); // overridden to draw this view  virtual BOOL PreCreateWindow(CREATESTRUCT& cs);  protected:  virtual BOOL OnPreparePrinting(CPrintInfo\* pInfo);  virtual void OnBeginPrinting(CDC\* pDC, CPrintInfo\* pInfo);  virtual void OnEndPrinting(CDC\* pDC, CPrintInfo\* pInfo);  // Implementation  public:  virtual ~CUP\_8View();  #ifdef \_DEBUG  virtual void AssertValid() const;  virtual void Dump(CDumpContext& dc) const;  #endif  protected:  // Generated message map functions  protected:  DECLARE\_MESSAGE\_MAP()  public:  afx\_msg void OnCircle();  afx\_msg void OnRect();  afx\_msg void OnClean();  afx\_msg void OnLButtonDown(UINT nFlags, CPoint point);  afx\_msg void OnMouseMove(UINT nFlags, CPoint point);  afx\_msg void OnLButtonUp(UINT nFlags, CPoint point);  };  #ifndef \_DEBUG // debug version in UP\_8View.cpp  inline CUP\_8Doc\* CUP\_8View::GetDocument() const  { return reinterpret\_cast<CUP\_8Doc\*>(m\_pDocument); }  #endif | **Файл UP\_8View.cpp**  // UP\_8View.cpp : implementation of the CUP\_8View class  //  #include "stdafx.h"  // SHARED\_HANDLERS can be defined in an ATL project implementing preview, thumbnail  // and search filter handlers and allows sharing of document code with that project.  #ifndef SHARED\_HANDLERS  #include "UP\_8.h"  #endif  #include "UP\_8Doc.h"  #include "UP\_8View.h"  #ifdef \_DEBUG  #define new DEBUG\_NEW  #endif  Circle circ(620, 340, 620, 380);  Rectangles rectan(580, 300, 660, 380);  bool cflag = false, rflag = false, cActive = false, rActive = false;  // CUP\_8View  IMPLEMENT\_DYNCREATE(CUP\_8View, CView)  BEGIN\_MESSAGE\_MAP(CUP\_8View, CView)  // Standard printing commands  ON\_COMMAND(ID\_FILE\_PRINT, &CView::OnFilePrint)  ON\_COMMAND(ID\_FILE\_PRINT\_DIRECT, &CView::OnFilePrint)  ON\_COMMAND(ID\_FILE\_PRINT\_PREVIEW, &CView::OnFilePrintPreview)  ON\_COMMAND(ID\_32775, &CUP\_8View::OnCircle)  ON\_COMMAND(ID\_32776, &CUP\_8View::OnRect)  ON\_COMMAND(ID\_32777, &CUP\_8View::OnClean)  ON\_WM\_LBUTTONDOWN()  ON\_WM\_MOUSEMOVE()  ON\_WM\_LBUTTONUP()  END\_MESSAGE\_MAP()  // CUP\_8View construction/destruction  CUP\_8View::CUP\_8View()  {  // TODO: add construction code here  }  CUP\_8View::~CUP\_8View()  {  }  BOOL CUP\_8View::PreCreateWindow(CREATESTRUCT& cs)  {  // TODO: Modify the Window class or styles here by modifying  // the CREATESTRUCT cs  return CView::PreCreateWindow(cs);  }  // CUP\_8View drawing  void CUP\_8View::OnDraw(CDC\* pDC)  {  CUP\_8Doc\* pDoc = GetDocument();  ASSERT\_VALID(pDoc);  if (!pDoc)  return;  if (cflag)  {  pDC->Ellipse(circ.getLXY(0) - (int)circ.getR(), circ.getLXY(1) + (int)circ.getR(), circ.getLXY(0) + (int)circ.getR(), circ.getLXY(1) - (int)circ.getR());  }  if (rflag)  {  pDC->Rectangle(rectan.getLXY(0), rectan.getLXY(1), rectan.getRXY(0), rectan.getRXY(1));  }  }  // CUP\_8View printing  BOOL CUP\_8View::OnPreparePrinting(CPrintInfo\* pInfo)  {  // default preparation  return DoPreparePrinting(pInfo);  }  void CUP\_8View::OnBeginPrinting(CDC\* /\*pDC\*/, CPrintInfo\* /\*pInfo\*/)  {  // TODO: add extra initialization before printing  }  void CUP\_8View::OnEndPrinting(CDC\* /\*pDC\*/, CPrintInfo\* /\*pInfo\*/)  {  // TODO: add cleanup after printing  }  // CUP\_8View diagnostics  #ifdef \_DEBUG  void CUP\_8View::AssertValid() const  {  CView::AssertValid();  }  void CUP\_8View::Dump(CDumpContext& dc) const  {  CView::Dump(dc);  }  CUP\_8Doc\* CUP\_8View::GetDocument() const // non-debug version is inline  {  ASSERT(m\_pDocument->IsKindOf(RUNTIME\_CLASS(CUP\_8Doc)));  return (CUP\_8Doc\*)m\_pDocument;  }  #endif //\_DEBUG  // CUP\_8View message handlers  void CUP\_8View::OnCircle()  {  cflag = true;  InvalidateRect(0, TRUE);  }  void CUP\_8View::OnRect()  {  rflag = true;  InvalidateRect(0, TRUE);  }  void CUP\_8View::OnClean()  {  cflag = false;  rflag = false;  InvalidateRect(0, TRUE);  }  void CUP\_8View::OnLButtonDown(UINT nFlags, CPoint point)  {  if (cflag && abs(point.x - circ.getLXY(0)) <= circ.getR() && abs(point.y - circ.getLXY(1)) <= circ.getR())  {  cActive = true;  }  if (rflag && point.x >= rectan.getLXY(0) && point.y >= rectan.getLXY(1)  && point.x <= rectan.getRXY(0) && point.y <= rectan.getRXY(1))  {  rActive = true;  }  CView::OnLButtonDown(nFlags, point);  }  void CUP\_8View::OnMouseMove(UINT nFlags, CPoint point)  {  if (cflag && (nFlags & MK\_SHIFT) && (abs(point.x - circ.getRXY(0)) < 20 && abs(point.y - circ.getRXY(1)) < 20))  {  SetCursor(LoadCursor(NULL, IDC\_SIZENS));  ShowCursor(TRUE);  }  if (rflag && (nFlags & MK\_SHIFT) && (abs(point.x - rectan.getRXY(0)) < 20 && abs(point.y - rectan.getRXY(1)) < 20))  {  SetCursor(LoadCursor(NULL, IDC\_SIZENWSE));  ShowCursor(TRUE);  }  if (nFlags & MK\_LBUTTON)  {  if (nFlags & MK\_SHIFT)  {  if (cActive && abs(point.x - circ.getRXY(0)) < 20 && abs(point.y - circ.getRXY(1)) < 20)  {  circ.fAlteration(point.x - circ.getRXY(0), point.y - circ.getRXY(1));  InvalidateRect(0, TRUE);  }  if (rActive && abs(point.x - rectan.getRXY(0)) < 20 && abs(point.y - rectan.getRXY(1)) < 20)  {  rectan.fAlteration(point.x - rectan.getRXY(0), point.y - rectan.getRXY(1));  InvalidateRect(0, TRUE);  }  }  else  {  if (cActive)  {  circ.fMove(point.x - circ.getLXY(0), point.y - circ.getLXY(1));  InvalidateRect(0, TRUE);  }  if (rActive)  {  rectan.fMove(point.x - rectan.getLXY(0), point.y - rectan.getLXY(1));  InvalidateRect(0, TRUE);  }  }  }  CView::OnMouseMove(nFlags, point);  }  void CUP\_8View::OnLButtonUp(UINT nFlags, CPoint point)  {  cActive = false;  rActive = false;  CView::OnLButtonUp(nFlags, point);  } |

## Скриншоты:



Использование STL. Обработка строк String.

Преобразование С-строк в String;

Найти подстроку (начиная с начала и с конца);

Копирование;

Сравнение;

Замещение фрагмента строки на другой фрагмент;

Вставка в строку другую строку с заданной позиции;

Поиск символа (начиная с начала и с конца);

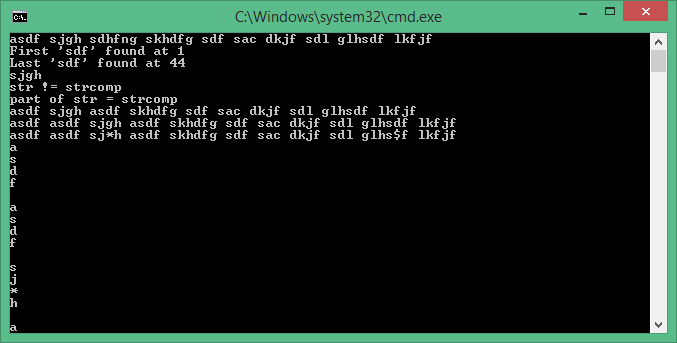
Использовать итераторы;

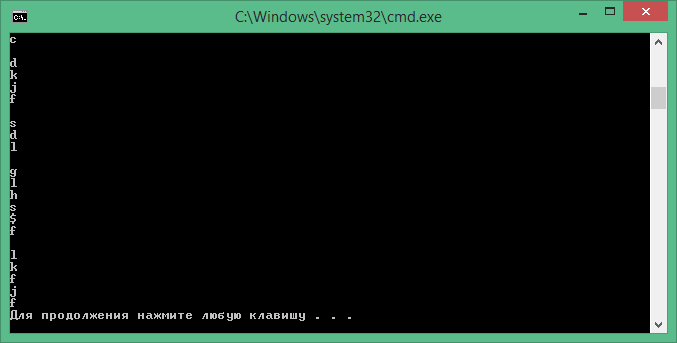
*Меню для программы можно не писать.*

## Код:

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  int main()  {  char cstr[] = "asdf sjgh sdhfng skhdfg sdf sac dkjf sdl glhsdf lkfjf";  string str, strcomp = "sdf";  size\_t found;  str = cstr;  cout << str << endl;  found = str.find("sdf");  if (found != str.npos)  {  cout << "First 'sdf' found at " << found << endl;  }  found = str.rfind("sdf");  if (found != str.npos)  {  cout << "Last 'sdf' found at " << found << endl;  }  strcpy(cstr, "");  found = str.copy(cstr, 4, 5);  cstr[found] = '\0';  cout << cstr << endl;  if (str.compare(strcomp) != 0)  {  cout << "str != strcomp" << endl;  }  if (str.compare(1, 3, strcomp) == 0)  {  cout << "part of str = strcomp" << endl;  }  str.replace(10, 7, str.substr(0, 5));  cout << str << endl;  str.insert(5, str.substr(0, 5));  cout << str << endl;  found = str.find\_first\_of("g");  if (found != str.npos)  {  str[found] = '\*';  }  found = str.find\_last\_of("d");  if (found != str.npos)  {  str[found] = '$';  }  cout << str << endl;  for (string::iterator it = str.begin(); it != str.end(); it++)  {  cout << \*it << endl;  }  return 0;  } |

## Скриншоты:





# Использование STL. Обработка контейнеров. Использование класса Vector.

добавление (в конец, середину по индексу),

удаление элементов,

изменение элементов;

сортировку (использовать алгоритм);

частичной сортировки (использовать алгоритм);

бинарного поиска в отсортированном векторе (использовать алгоритм);

сравнение (поиск вектора в векторе),

копирование (во второй вектор);

объединение векторов (использовать алгоритм);

вычисления суммы элементов (использовать алгоритм);

изменение размера вектора;

определение мах размера вектора, количество элементов в векторе, сколько в векторе может храниться до изменения размера вектора.

использовать итераторы

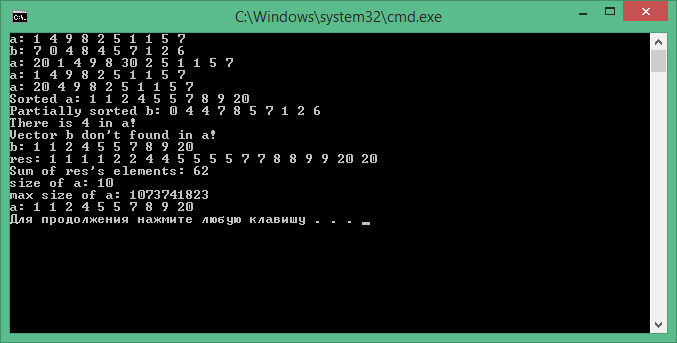
использовать итераторы вывода (ostream\_iterator).

*Меню для программы можно не писать.*

## Код:

|  |
| --- |
| #include <iostream>  #include <vector>  #include <algorithm>  #include <numeric>  #include <iterator>  using namespace std;  void print(vector<int>& vec)  {  for (vector<int>::iterator it = vec.begin(); it != vec.end(); it++)  {  cout << \*it << " ";  }  cout << endl;  }  int main()  {  vector<int> a, b, res;  vector<int>::iterator iter, iterat;  ostream\_iterator<int> outit(cout, " ");  for (int i = 0; i < 10; i++)  {  a.push\_back(rand() % 10);  b.push\_back(rand() % 10);  }  cout << "a: ";  print(a);  cout << "b: ";  print(b);  iter = a.begin();  iter = a.insert(iter, 20);  iter = a.insert(iter + 5, 30);  cout << "a: ";  print(a);  iter = a.begin();  a.erase(iter);  iter = a.begin();  a.erase(iter + 4);  cout << "a: ";  print(a);  iter = a.begin();  \*iter = 20;  cout << "a: ";  print(a);  sort(a.begin(), a.end());  cout << "Sorted a: ";  print(a);  iter = b.begin();  sort(iter, iter + 5);  cout << "Partially sorted b: ";  print(b);  iter = a.begin();  if (binary\_search(a.begin(), a.end(), 4))  {  cout << "There is 4 in a!\n";  }  else  {  cout << "There is no 4 in a!\n";  }  sort(a.begin(), a.end());  sort(b.begin(), b.end());  iter = find\_first\_of(a.begin(), a.end(), b.begin(), b.begin() + 5);  if (iter != a.end())  {  cout << "Vector b don't found in a!" << endl;  }  else  {  cout << "Vector b found in a!" << endl;  }  copy(a.begin(), a.end(), b.begin());  cout << "b: ";  print(b);  sort(a.begin(), a.end());  sort(b.begin(), b.end());  res.resize(a.size() + b.size());  merge(a.begin(), a.end(), b.begin(), b.end(), res.begin());  cout << "res: ";  print(res);  cout << "Sum of res's elements: " << accumulate(a.begin(), a.end(), 0) << endl;  cout << "size of a: " << a.size() << "\nmax size of a: " << a.max\_size() << endl;  cout << "a: ";  copy(a.begin(), a.end(), outit);  cout << endl;  return 0;  } |

## Скриншоты:



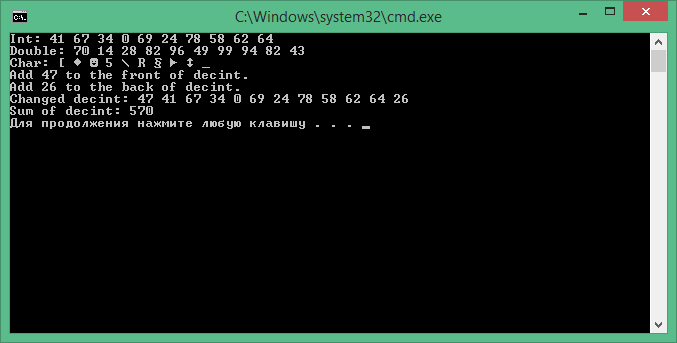
# Адаптеры контейнеров

Разработать адаптер контейнера deque на основе vector. Разработать функции добавления, печати, вычисления суммы элементов очереди. Написать тестовую программу для типов int, double, char.

## Код:

|  |  |
| --- | --- |
| **Файл Deque.h**  #ifndef DEQUE\_H  #define DEQUE\_H  #include <iostream>  #include <vector>  using namespace std;  template <class T>  class Deque  {  vector<T> data;  public:  Deque();  Deque(int size);  Deque(const Deque<T>& init);  ~Deque();  bool isEmpty();  T front();  T back();  void push\_front(T dat);  void push\_back(T dat);  void pop\_front();  void pop\_back();  size\_t size();  void print(ostream& out);  T sum();  };  template <class T>  Deque<T>::Deque()  {  data = vector<T>;  }  template <class T>  Deque<T>::Deque(int size)  {  for (int i = 0; i < size; i++)  {  data.push\_back(rand() % 100);  }  }  template <>  Deque<double>::Deque(int size)  {  for (int i = 0; i < size; i++)  {  data.push\_back((rand() % 1000) / 10);  }  }  template <class T>  Deque<T>::Deque(const Deque<T>& init)  {  data = init.data;  }  template <class T>  Deque<T>::~Deque()  {  while (!isEmpty())  {  pop\_back();  }  }  template <class T>  bool Deque<T>::isEmpty()  {  return data.empty();  }  template <class T>  T Deque<T>::front()  {  return data.front();  }  template <class T>  T Deque<T>::back()  {  return data.back();  }  template <class T>  void Deque<T>::push\_front(T dat)  {  vector<T>::iterator iter;  iter = data.begin();  data.insert(iter, dat);  }  template <class T>  void Deque<T>::push\_back(T dat)  {  data.push\_back(dat);  }  template <class T>  void Deque<T>::pop\_front()  {  vector<T>::iterator iter;  iter = data.begin();  data.erase(iter);  }  template <class T>  void Deque<T>::pop\_back()  {  data.pop\_back();  }  template <class T>  size\_t Deque<T>::size()  {  return data.size();  }  template <class T>  void Deque<T>::print(ostream& out)  {  for (auto i = data.begin(); i != data.end(); i++)  {  out << \*i << ' ';  }  out << endl;  }  template <class T>  T Deque<T>::sum()  {  T res = 0;  for (auto i = data.begin(); i != data.end(); i++)  {  res += \*i;  }  return res;  }  #endif | **Файл Source.cpp**  #include <iostream>  #include "Deque.h"  #include <deque>  using namespace std;  int main()  {  Deque<int> deqint(10);  Deque<double> deqdouble(10);  Deque<char> deqchar(10);  int tmp;  cout << "Int: ";  deqint.print(cout);  cout << "Double: ";  deqdouble.print(cout);  cout << "Char: ";  deqchar.print(cout);  tmp = rand() % 100;  cout << "Add " << tmp << " to the front of decint." << endl;  deqint.push\_front(tmp);  tmp = rand() % 100;  cout << "Add " << tmp << " to the back of decint." << endl;  deqint.push\_back(tmp);  cout << "Changed decint: ";  deqint.print(cout);  cout << "Sum of decint: " << deqint.sum() << endl;  } |

## Скриншоты:



Белорусский государственный университет

Факультет прикладной математики и информатики

Кафедра технологий программирования

Жук Валерий Викторович

**РЕФЕРАТ**

**на тему «Рисование объектов. Вставка картинок. WinApi»**

студента 1 курса 6 группы

**Преподаватель**

***Сатолина Анна Викторовна***

Ассистент кафедры технологий программирования ФПМИ

Минск, 2016

Рисование окна осуществляется при обработке сообщения WM\_PAINT, обработка которого в общем случае выглядит так:

**case WM\_PAINT :**

**hDC=BeginPaint(hWnd, &ps);  
//здесь можно вставить какие-нибудь функции рисования  
ValidateRect(hWnd, NULL); // (или InvalidateRect(hWnd, NULL)) обновляем окно  
EndPaint(hWnd, &ps); //заканчиваем рисовать  
break;**

Где hDC - экземпляр контекста устройства, ps - структуре информация об окне для прорисовки(PAINTSTRUCT), hWnd - дескриптор окна.

Теперь рассмотрим некоторые стандартные функции рисования:

**1. Вывод точки.**

SetPixel устанавливает заданный цвет в точке с указанными координатами:  
***COLORREF SetPixel(HDC hDC, int x, int y, COLORREF crColor);***  
Функция GetPixel соответственно возвращает цвет в заданных координатах.  
***COLORREF Getpixel(hDC, int x, int y);***  
  
**2. Рисование линий.**  
***BOOL LineTo(hDC, int x, int y);***  
Функция рисует линию от текущей позиции до места, указанного в аргументах. Чтобы изменить тип линии (толщину, стиль)- меняется тип пера. Но об этом позже.   
  
Так как в отличие от многих других подходов, в GDI нет функции рисования линии от одного указанного места до другого, её можно создать самому. Она будет соединять линией точки с координатами: x1,y1 и x2,y2.  
  
***BOOL Line(HDC hdc, int x1, int y1, int x2, int y2)  
{  
MoveToEx(hdc, x1, y1, NULL); //сделать текущими координаты x1, y1  
return LineTo(hdc, x2, y2);  
}***  
  
**3. Дуга.**  
**BOOL Arc(hDC, int left, int top, int right, int bottom, int x1, int y1, int x2, int y2);**  
Первые четыре аргумента - левый верхний и правый нижний углы прямоугольника, в который вписан эллипс. Остальные значения - координаты точек, от которых будут проведены прямые к центру эллипса. В местах пересечения первой и второй прямой с радиусом эллипса, начинается и кончается дуга.  
  
**4. Прямоугольник.**

По умолчанию прозрачный, а вообще, тип его заливки определяется текущей кистью. По умолчанию она тоже прозрачная.   
**BOOL Rectangle(hDC, int left, int top, int right, int bottom); //аргументы - это коордианты левого верхнего и правого нижнего углов**  
**5. Закруглённый прямоугольник.**   
**BOOL RoundRect(hDC, int left, int top, int right, int bottom, int width, int height);**

Первые пять параметров совпадают с параметрами предыдущей фукнции. Далее width и height задают ширину и высоту эллипса, дуги которого ограничивают прямоугольник.   
  
**6. Кисти.**

Кисть закрашивает определённую область в определённый цвет. В зависимости от кисти, она может делать это в полосочку, в клеточку, по диагонали и т.д.  
Есть два способа объявить кисть. Первый - задать сплошную заливку, второй - указать стиль. Для этого существуют соответственно функции: **CreateSoldBrush()** и **CreateHatchBrush().**  
  
Пример:   
**HBRUSH hBrush; //создаём объект-кисть  
CreateSolidBrush(RGB(45,145,67)); //задаём сплошную кисть, закрашенную цветом RGB  
SelectObject(hdc, hBrush); //делаем кисть активной**  
  
А вот как объявить не сплошную кисть:  
**CreateHatchBrush(int fnStyle, RGB(r,g,b));**  
  
Аргумент fnStyle принимает ряд константных значений:  
HS\_DIAGONAL - штрихует по диагонали  
HS\_CROSS - клеточка  
HS\_DIAGCROSS - диагональная сетка  
HS\_FDIAGONAL - по диагонали в другую сторону  
HS\_HORIZONTAL - горизонтальная "тельняшка"  
HS\_VERTICAL - вертикальный "забор"  
  
**HBRUSH hBrush1;  
CreateHatchBrush(int fnStyle, RGB(r,g,b));  
SelectObject(hdc, hBrush1); //делаем кисть активной**  
  
Если после выбора кисти вызвать функцию рисования фигуры, то мы получим закрашенную(заштрихованную и т.д. в зависимости от выбора стиля кисти) фигуру.

**7. Перья.**

Линия может быть жирной и тонкой, прерывистой и штрих-пунктирной. Всё предусмотрено.  
  
**HPEN hPen; //Объявляется кисть  
CreatePen(fnPenStyle, int width, RGB(r,g,b)); //Создаётся объект  
SelectObject(hdc, hPen); //Объект делается текущим**  
fnStyle может принимать следующие значения:  
PS\_SOLD - сплошная  
PS\_DASH - состоящая из точек  
PS\_DOT - состоящая из тире  
PS\_DASHDOT - "точка-тире"  
PS\_DASHDOTDOT - "тире-точка-точка-тире"  
PS\_NULL - невидимая  
PS\_INSIDEFRAME - обводка замкнутых фигур

Важно понять, что можно создавать хоть 10 перьев с помощью CreatePen, но применить в данный момент времени можно только 1 из них. Для этого и нужен SelectObject, чтобы окно поняло какую кисть в настоящий момент мы используем.  
  
**8. Закрашенный прямоугольник.**  
int FillRect(HDC hDC, CONST RECT \*lprc, HBRUSH hbr);  
lprc - закрашиваемый прямоугольник типа RECT.  
hbr - кисть  
  
Вот пример-фрагмент WM\_PAINT:  
  
**RECT r; //объявляем экзмепляр структуры RECT - координаты прямоугольника.  
r.left=100; //левый верхний угол  
r.top=100;   
r.right=200; //правый нижний  
r.right=300;**

**FillRect(hdc, &r, (HBRUSH)CreateSolidBrush(RGB(255,0,0))); //Заполняем прямоугольник**

**7. Прямоугольная рамка.  
  
int FrameRect(HDC hDC, CONST RECT \*lprc, HBRUSH hbr);**Применение аналогично предыдущей. **8. Инверсия значения цветов точек в заданной области.  
BOOL InvertRect(HDC hDC, CONST RECT \*lprc);** **9. Эллипс.  
BOOL Ellipse(HDC hdc, int x1, int y1, int x2, int y2);**координаты - это прямоугольник, в который вписывается эллипс **10. Хорда (сегмент эллипса).  
BOOL Chord(HDC hDC, int left, int top, int right, int bottom, int x1, int y1, int x2, int y2);**

Параметры аналогичны Arc.Функция соединяет хордой точки начала и конца дуги эллипса и закрашивает выделенный сегмент текущей кистью. **11. Сектор эллипса.  
BOOL Pie(HDC hDC, int left, int top, int right, int bottom, int x1, int y1, int x2, int y2);** **12. Многоугольник.**

Есть много функций рисования мноугольников. Мы рассмотрим две. Рисования от вершины к вершине и рисования отрезками: **PolyDraw** оперирует вершинами**:  
  
POINT poly[8];  
BYTE polytype[8];  
  
poly[0].x=375; //координаты первой вершины  
poly[0].y=375;  
....//и так заполняем координаты всех восьми вершин  
poly[7].x=400; //координаты восьмой вершины  
poly[7].y =400;  
  
polytype[0]=PT\_LINETO;  
... //другой массив содержит режим рисования  
polytype[7]=PT\_LINETO;   
  
PolyDraw(hdc, poly, polytype, 8); //рисование многоугольника**Функция**Polyline**рисует набором отрезков: **POINT poly[4];**

**poly[0].x =10;  
poly[0].y =30;**

**poly[1].x =30;  
poly[1].y =30;**

**poly[2].x =30;  
poly[2].y =60;**

**poly[3].x=10;  
poly[3].y=30;**

**Polyline(hdc, poly , 4);**

Вывод картинки:

**HDC hDC, hCompatibleDC;**

**PAINTSTRUCT PaintStruct;**

**HANDLE hBitmap, hOldBitmap; // дескриптор объекта**

**RECT Rect;**

**BITMAP Bitmap; // дескриптор растрового изображения, используется для работы с изображениями**

**switch(Message)**

**{**

**case WM\_PAINT:**

**hDC = BeginPaint(hWnd, &PaintStruct); // Получаем контекст устройства.**

**hBitmap = LoadImage(NULL, L"MSDOGS.BMP", IMAGE\_BITMAP,**

**0, 0, LR\_LOADFROMFILE); // Загружаем bitmap, который будет отображаться в окне, из фаила.**

**GetObject(hBitmap, sizeof(BITMAP), &Bitmap); // Получаем размероность загруженного bitmap'a.**

**hCompatibleDC = CreateCompatibleDC(hDC); //Создаем совместный с контекстом окна контекст в памяти.**

**hOldBitmap = SelectObject(hCompatibleDC, hBitmap); //Делаем загруженный из фаила bitmap текущим в совместимом контексте.**

**GetClientRect(hWnd, &Rect); //Определяем размер рабочей области окна.**

**StretchBlt(hDC, 0, 0, Rect.left, Rect.bottom, hCompatibleDC, 0, 0, Bitmap.bmWidth,**

**Bitmap.bmHeight, SRCCOPY); //Копируем bitmap с совместимого на основной контекст с масштабированием.**

**SelectObject(hCompatibleDC, hOldBitmap); //Вновь делаем старый bitmap текущим.**

**DeleteObject(hBitmap); //Удаляем загруженный с диска bitmap.**

**DeleteDC(hCompatibleDC); //Удаляем совместный контекст.**

**EndPaint(hWnd, &PaintStruct); //Освобождаем основной контекст, завершая перерисовку рабочей области окна.**

**break;**

Белорусский государственный университет

Факультет прикладной математики и информатики

Кафедра технологий программирования

Жук Валерий Викторович

**РЕФЕРАТ**

**на тему «Контейнеры map и multimap»**

студента 1 курса 6 группы

**Преподаватель**

***Сатолина Анна Викторовна***

Ассистент кафедры технологий программирования ФПМИ

Минск, 2016

map — отсортированный ассоциативный контейнер, который содержит пары ключ-значение с неповторяющимися ключами. Порядок ключей задаётся функцией сравнения Compare. Операции поиска, удаления и вставки имеют логарифмическую сложность. Данный тип, как правило, реализуется как красно-чёрное дерево.

**template<  
 class Key,  
    class T,  
    class Compare = std::less<Key>,  
    class Allocator = std::allocator<std::pair<const Key, T> >**

**> class map;**

**Key -** тип данных ключа для сохранения в сопоставлении.

**Type -** тип данных элемента для сохранения в сопоставлении.

**Traits -** тип, предоставляющий объект функции, который может сравнить два значения элемента как ключи сортировки, чтобы определить их относительный порядок в сопоставлении.Этот аргумент является необязательным, и в качестве значения по умолчанию используется бинарный предикат less<*Key*>.

В C++ 14 вы можете включить разнородный поиск, указав предикат std::less<>, не имеющий параметров типа.

**Allocator -** тип, представляющий сохраненный объект распределителя, который инкапсулирует сведения о выделении и освобождении памяти для сопоставления.Этот аргумент является необязательным, и значением по умолчанию является allocator<pair<const*Key, Type*> >.

Также в контейнере map не существует элементов с одинаковыми ключами.

Контейнер map имеет все типы итераторов, как прямые так и обратные. Для работы с итераторами в контейнере map предусмотрены следующие методы:

**begin()** – возвращает итератор указывающий на первый элемент контейнера map.

**end()** – возвращает итератор указывающий на элемент после последнего в контейнере map.

**rbegin()** – реверсивный begin().

**rend()** – реверсивный end().

**cbegin(), cend(), crbegin(), crend() –** идентичны предыдущим методам, но используются в константном map.

Методы для работы с ёмкостью контейнера map:

**empty()** – возвращает true если контейнер пуст и false в других случаях.

**size()** – возвращает размер контейнера. Возвращаемый тип size\_type(переопределённый unsigned int).

**max\_size()** – возвращает максимальное количество элементов которое может быть помещено в контейнер. Возвращаемый тип size\_type(переопределённый unsigned int).

Доступ к элементам контейнера map:

**mapped\_type& operator[] (key\_type&& k)** и **mapped\_type& at (const key\_type& k)** осуществляют доступ к элементу контейнера по ключу.

Модификаторы:

**insert:**

**pair<iterator,bool> insert (const value\_type& val)**

**pair<iterator,bool> insert (P&& val) –** производит вставку элемента в контейнер, сравнивая ключ.

**iterator insert (const\_iterator position, const value\_type& val)**

**iterator insert (const\_iterator position, P&& val) –** производит вставку элемента в контейнер с попыткой вставки его в определённое место контейнера.

**void insert (InputIterator first, InputIterator last)** – производит вставку последовательности элементов в контейнер.

**void insert (initializer\_list<value\_type> il); -** производит вставку объекта в контейнер.

**erase:**

**iterator erase (const\_iterator position) –** производит удаление элемента находящегося в заданной позиции.

**size\_type erase (const key\_type& k) –** производит удаление элементов по ключу.

**iterator erase (const\_iterator first, const\_iterator last) –** удаляет последовательность элементов контейнера.

**swap:**

**void swap (map& x) –** меняет местами содержимое контейнеров.

**clear:**

**void clear()** – очищает контейнер, удаляя всего его элементы.

**emplace:**

**pair<iterator,bool> emplace (Args&&... args) –** производит вставку элемента в контейнер, если ключ элемента уникален, т.е. отсутствует в контейнере.

**iterator emplace\_hint (const\_iterator position, Args&&... args) –** аналогичен обычному emplace, только производит попытку вставки элемента в указанную позицию контейнера.

Обозреватели:

**key\_compare key\_comp()** – возвращает объект сравнения.

**value\_compare value\_comp() -** объект сравнения для значений элементов.

Операции:

**iterator find (const key\_type& k) –** производит поиск элемента в контейнере по ключу.

**size\_type count (const key\_type& k)** – подсчитывает количество элементов с одинаковым ключом.(в map возвращает только 1 или 0, в multimap – любое число больше или равное нуля).

**iterator lower\_bound (const key\_type& k) -** возвращает итератор, указывающий на первый элемент в сопоставлении с ключом, который больше или равен указанному ключу.

**iterator upper\_bound (const key\_type& k) -** возвращает итератор, указывающий на первый элемент в сопоставлении с ключом, который больше указанного ключа.

**pair<iterator,iterator> equal\_range (const key\_type& k)** - возвращает пару итераторов. Первый итератор в паре указывает на первый элемент в map с ключом, который больше указанного ключа. Второй итератор в паре указывает на первый элемент в map с ключом, который больше или равен данному ключу.

**allocator\_type get\_allocator() –** возвращает allocator.

Контейнерmultimap идентичен map, за исключением того, что позволяет дублировать ключи, в связи с чем в нём недоступен operator[].