# GRAFIKA KOMPUTEROWA I ANIMACJA

**Dr inż. Sławomir Samolej**
**D102 C, tel: 865 1766,**
**email: ssamolej@prz-rzeszow.pl**
**www: mail.prz-rzeszow.pl/~ssamolej**

**Wymagania wstępne:**

- **Znajomość programowania w języku C**

# Program zajęć

**Wykład:**

1. **Wprowadzenie do grafiki komputerowej, programowania w Windows,**
2. **Formaty plików graficznych,**
3. **Standard OpenGL – aplikacja Windows+ OpenGL, prymitywy graficzne,**
4. **Standard OpenGL – transformacje przestrzenne,**
5. **Standard OpenGL – oświetlenie sceny,**
6. **Standard OpenGL – teksturowanie,**
7. **Standard OpenGL – krzywe i powierzchnie, grafika komputerowa w grach.**

**Laboratorium:**

1. **Programowane Windows**
2. **Formaty plików graficznych**
3. **Tworzenie siatek**
4. **Transformacje przestrzenne**
5. **Oświetlenie sceny**
6. **Teksturowanie**
7. **Krzywe i powierzchnie parametryczne**

# Literatura


1. Richard S. Wright jr, Michael Sweet: „ OpenGL™ Księga eksperta”, Helion 1999.
2. Kevin Hawkins, Dave Astle: „OpenGL. Programowanie gier”, Helion 2003.
3. Charles Petzold: „Programowanie Windows”, wydawnictwo RM, 1999.
4. Sławomir Samolej: „Grafika komputerowa” – materiały laboratoryjne.
5. „OpenGL Programming Guide” (www.opengl.org).
6. „OpenGL Reference Manual” (www.opengl.org).
7. Mason Woo, Jackie Neider, Tom Davis: „OpenGL™ Programming Guide”, Addison-Wesley 1996.
8. Edward Angel: „Interactive Computer Graphics: A Top-Down Approach with OpenGL™”, Addison-Wesley 1997.
9. Ron Fosner: „OpenGL™ Programming for Windows and Windows NT”, Addison-Wesley Developers Press, 1997.
10. Jan Zabrodzki (red.): „Grafika komputerowa metody i narzędzia”, WNT 1994

# Przydatne odnośniki

- http://mail.prz-rzeszow.pl/~ssamolej

**OpenGL:**

- http://www.opengl.org
- http://www.sgi.com/software/opengl
- http://www.codeproject.com/opengl

**Programowanie gier:**

- http://www.gamedev.net
- http://nehe.gamedev.net
- http://www.gametutorials.com

**(po polsku):**

- http://warsztat.pac.pl

# Narzędzia programowe

- **Biblioteka OpenGL (wbudowana w Windows – bezpłatna)**
- **Dowolny kompilator C/C++ na platformę Windows**
  **(stosowany będzie Microsoft Visual C++ 6.0)**
- **Istnieją także implementacje OpengGL na:**
  - **Linux**
  - **Unix**

# Warunki uzyskania zaliczenia

- **Uczestnictwo w zajęciach laboratoryjnych.**
- **Zaliczenie połowy kolokwiów z zajęć laboratoryjnych.**

# Definicje i zastosowania

***GRAFIKA KOMPUTEROWA**, inform.* **metody wykonywania i przetwarzania obrazów graf. za pomocą programów komputerowych; obejmuje m.in. algorytmy umożliwiające graf. przedstawianie obiektów trójwymiarowych, ich przekształcenia i animację. (Encyklopedia PWN)**

**Zastosowania grafiki komputerowej:**

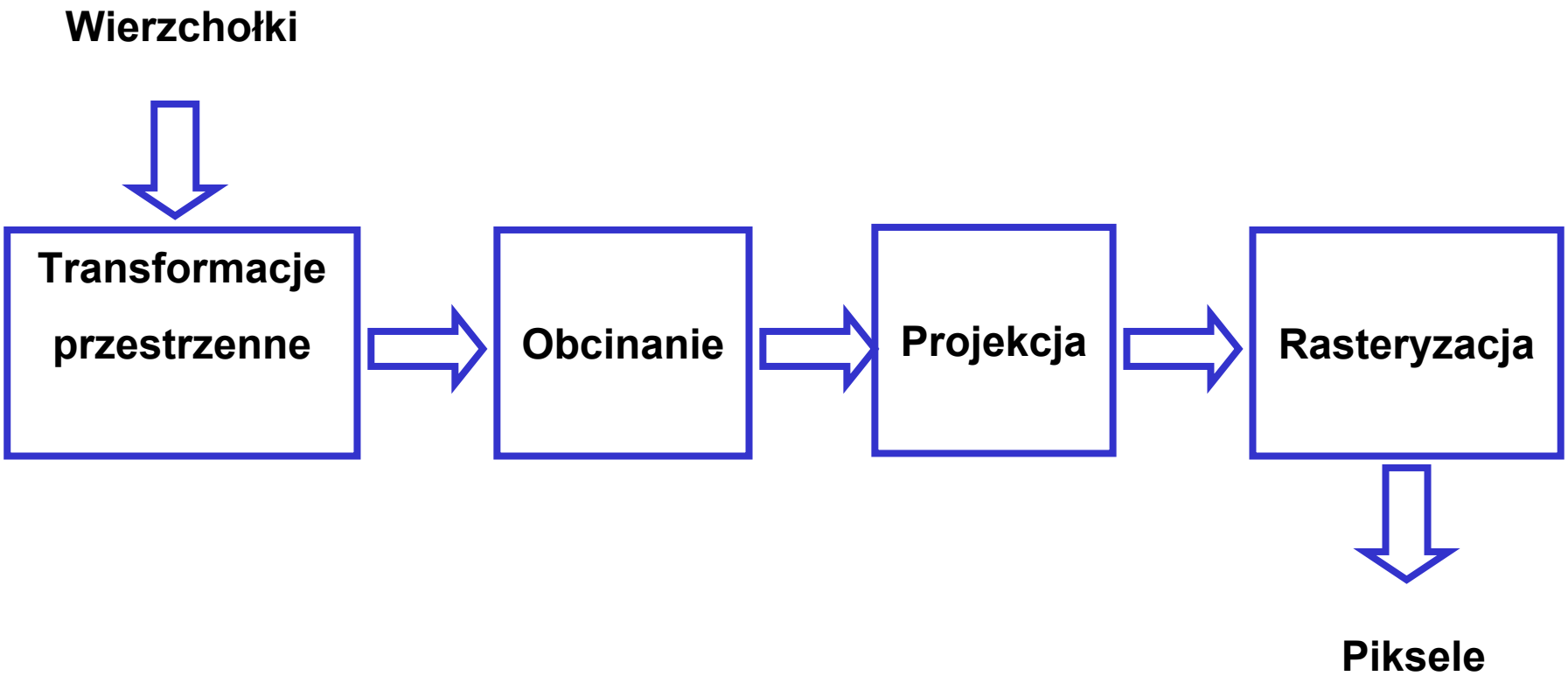
- **Wyświetlanie informacji (medycyna, nauka, przemysł, zarządzanie),**
- **Projektowanie (aplikacje CAD),**
- **Symulacje (np. symulatory lotu, gry komputerowe, rzeczywistość wirtualna),**
- **Interfejsy użytkownika.**

# Grafika rastrowa

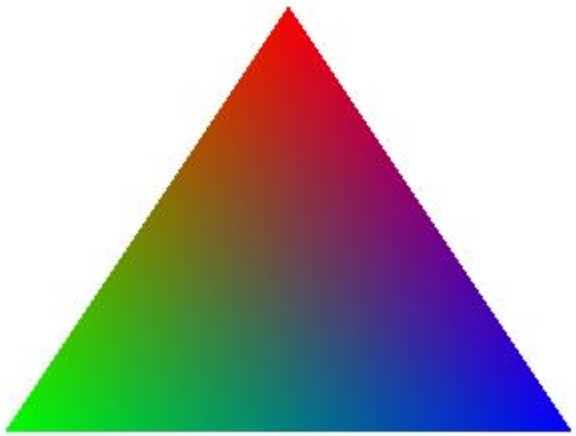
- **Obraz składa się z tablicy (rastra) elementów lub pikseli**
- **Każdy piksel odpowiada pewnemu obszarowi obrazu**

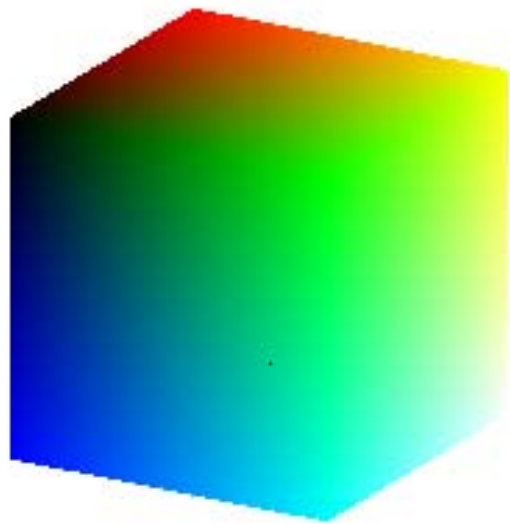
- **Piksele są gromadzone w pamięci obrazu (ang. frame buffer)**
- **Głębokość pamięci obrazu określa się przez ilość bitów, które służą do opisania koloru piksela**
  **(1 – obraz czarno – biały; 8 – 256 kolorów; 32 – ponad 4 miliony kolorów)**
- **Rozdzielczość określa ilość pikseli zawartych w pamięci obrazu i decyduje o poziomie szczegółowości obrazu.**

# Łańcuch przekształceń odwzorowujących przestrzeń na urządzeniu rastrowym



# Odwzorowywanie kolorów – Model RGB



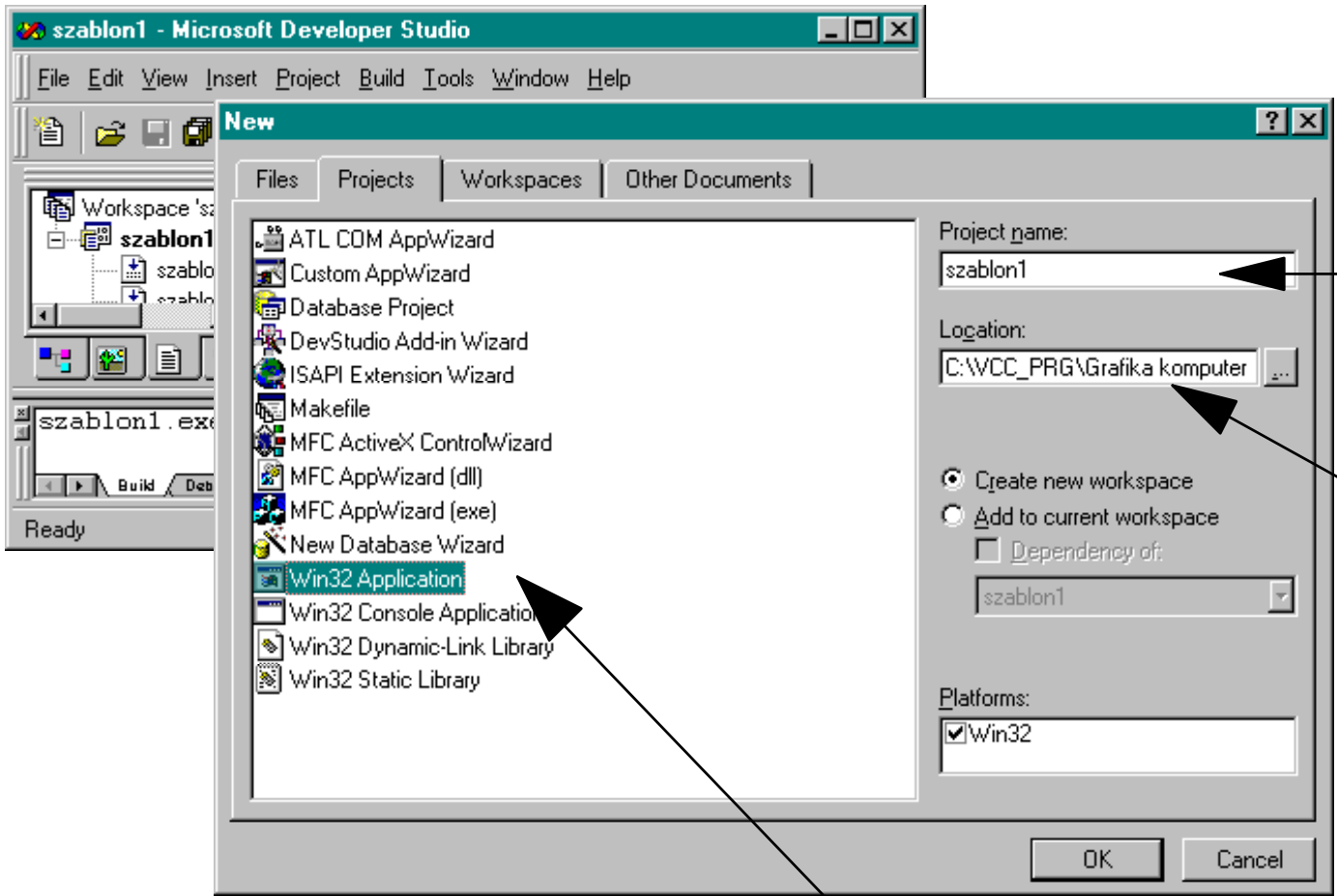


# Programowanie Windows - etapy tworzenia aplikacji

1. **Utworzenie nowego folderu przeznaczonego na pliki należące do projektu wraz z plikiem projektowym zawierającym dane na temat kompilowania i konsolidowania programu.**
2. **Wykorzystanie środowiska programowania do utworzenia zasobów aplikacji (menu, ikony, okna dialogowe, paski narzędzi, bitmapy, klawisze skrótu, zasoby tekstowe) i dołączenie pliku z zasobami do pliku projektowego.**
3. **Utworzenie plików źródłowych z funkcjami sterującymi aplikacją oraz dołączenie ich do pliku projektowego.**
4. **Kompilacja, konsolidacja i uruchomienie programu.**

# Programowanie Windows - tworzenie projektu

1. **File -> New**
2. **Zakładka „Projects”, wybrać Win 32 Application**



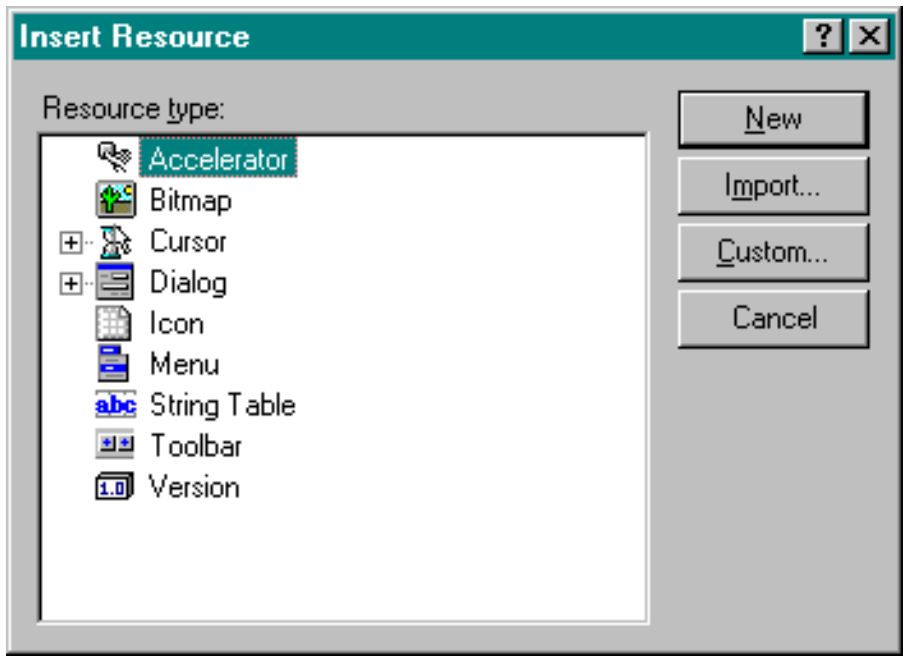

Nazwa nowego projektu

Lokalizacja folderu z tworzoną nową aplikacją

Wybór typu tworzonej aplikacji

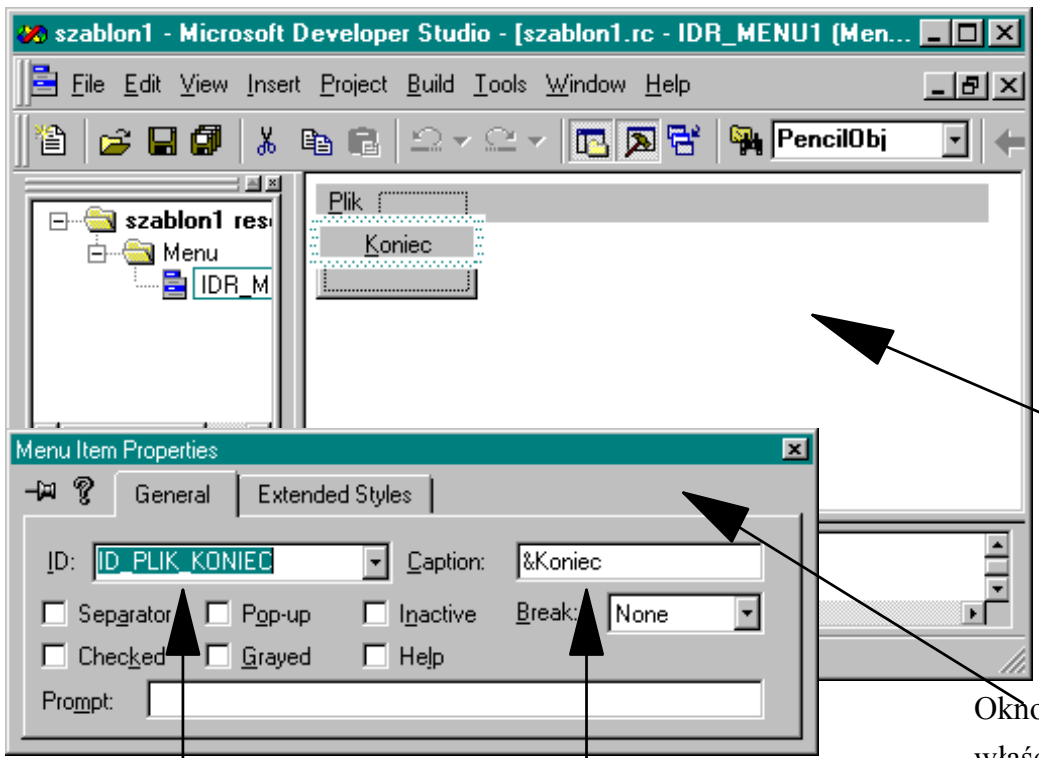
# Programowanie Windows - generowanie zasobów

1. **Insert -> Resource, wybrać menu**



# Programowanie Windows - generowanie zasobów

2. **Zdefiniować kolejne opcje menu, zwrócić uwagę na identyfikatory**
3. **Zachować plik *script1.rc* w folderze projektu**




Okno edytora menu

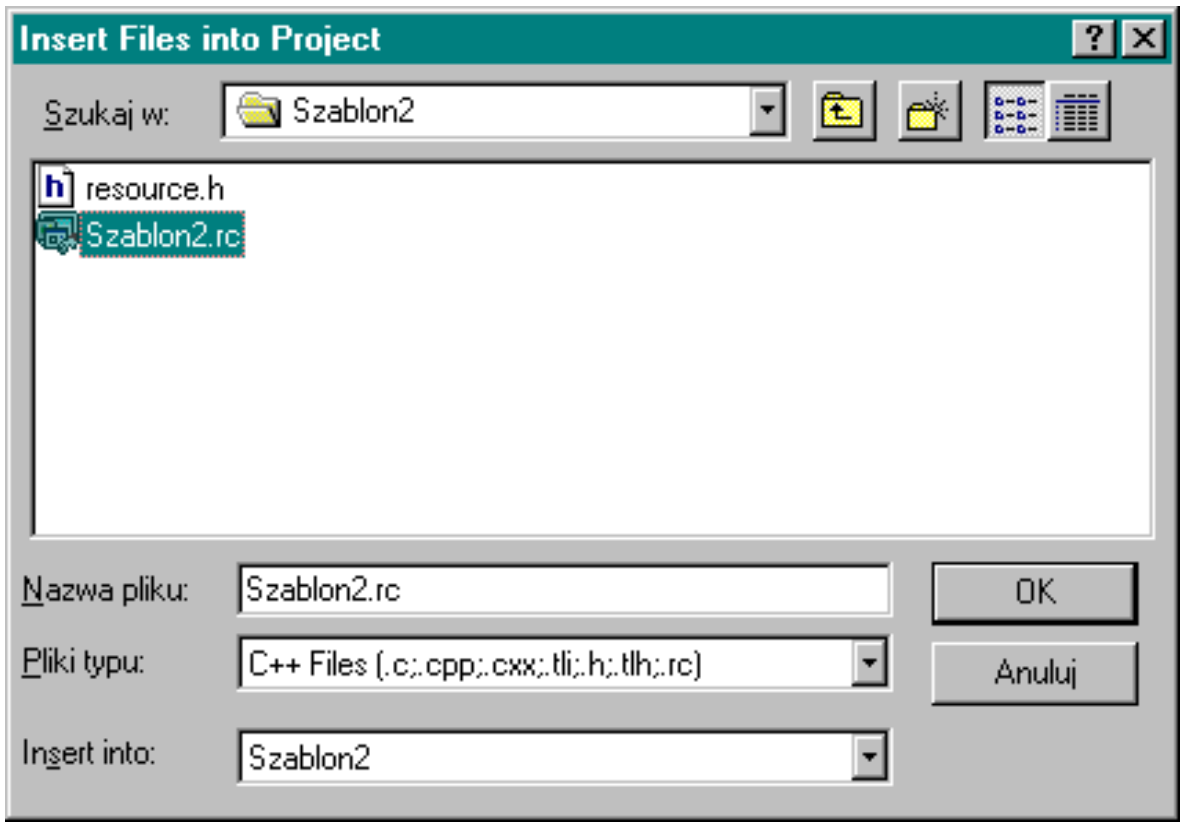
Okno dialogowe właściwości opcji menu

Identyfikator opcji menu

Tekst wyświetlany w polu opcji

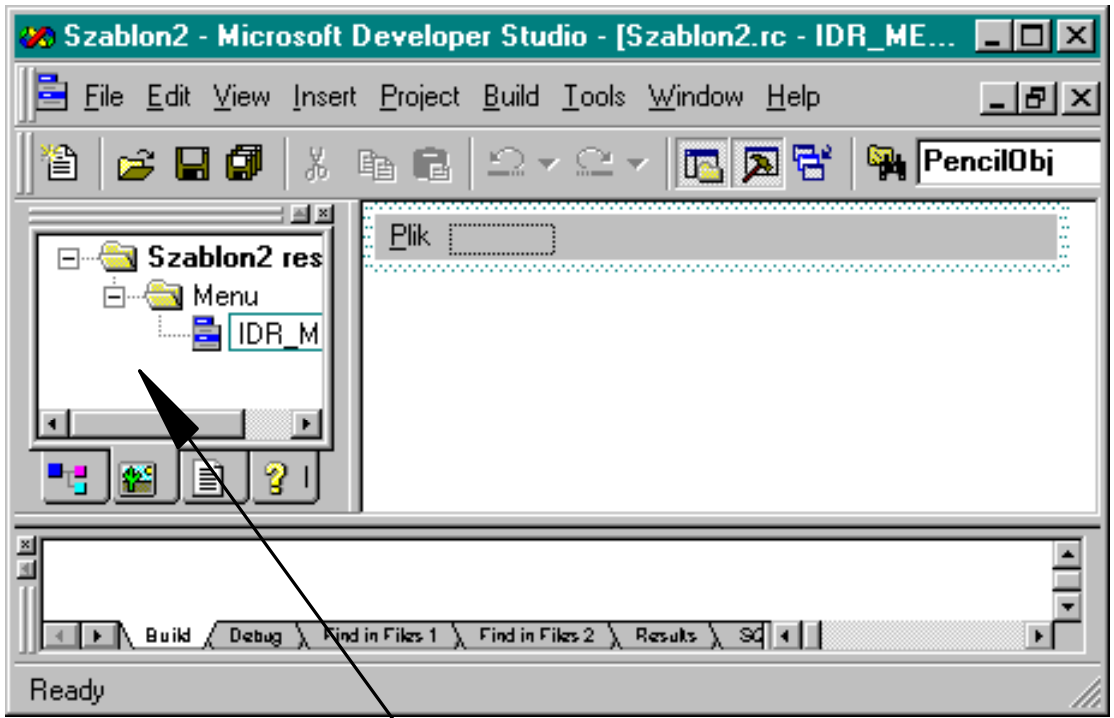
# Programowanie Windows - generowanie zasobów

4. **Dołączyć plik ze zdefiniowanymi zasobami do projektu (Project -> Add To Project -> Files, wybrać *Script1.rc***



# Programowanie Windows - generowanie zasobów

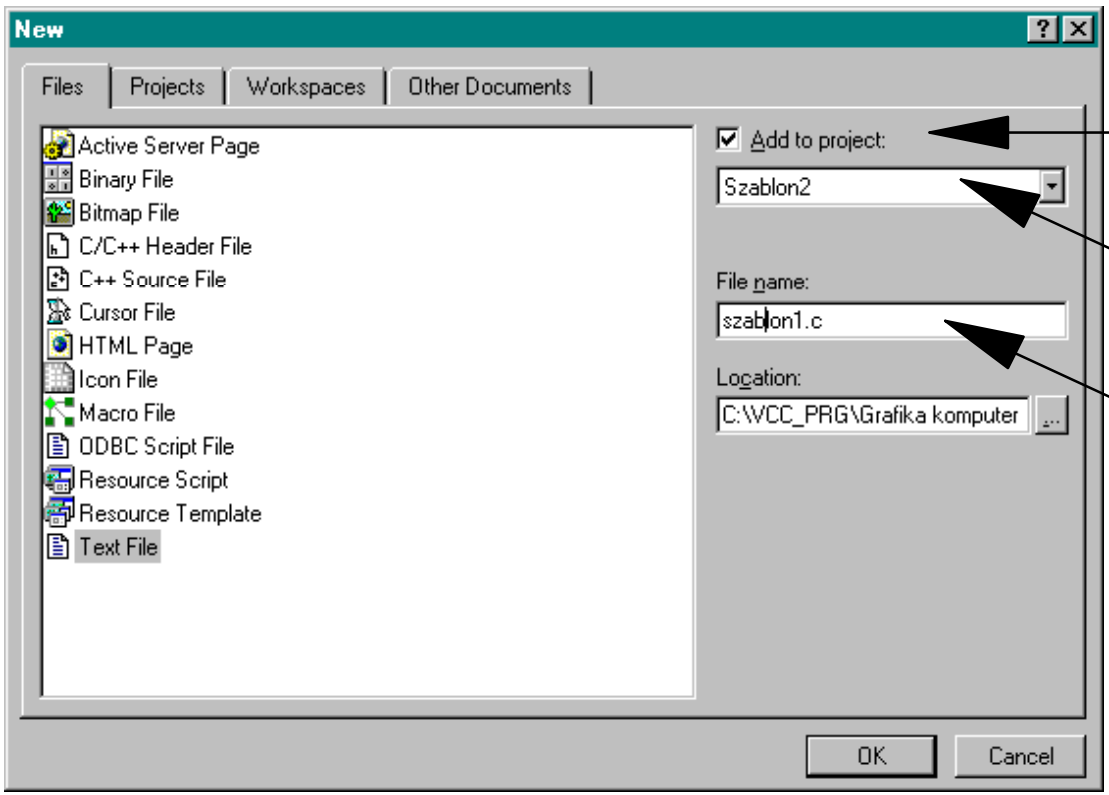
## 5. W oknie projektu dostępne są zasoby




Okno "Workspace"

# Programowanie Windows - tworzenie plików źródłowych

1. File -> New
2. Zakładka „Files”, wybrać C++ Source File
3. Podać nazwę pliku oraz wskazać, że ma być dołączony do projektu

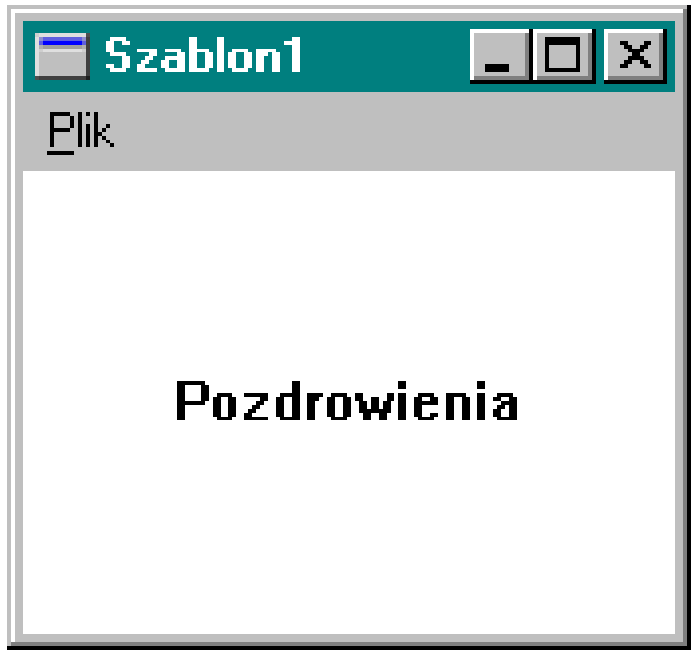



Pole wyboru zezwolenia na dołączenie tworzonego pliku tekstowego do projektu

Nazwa projektu, do kórego będzie włączony tworzony plik

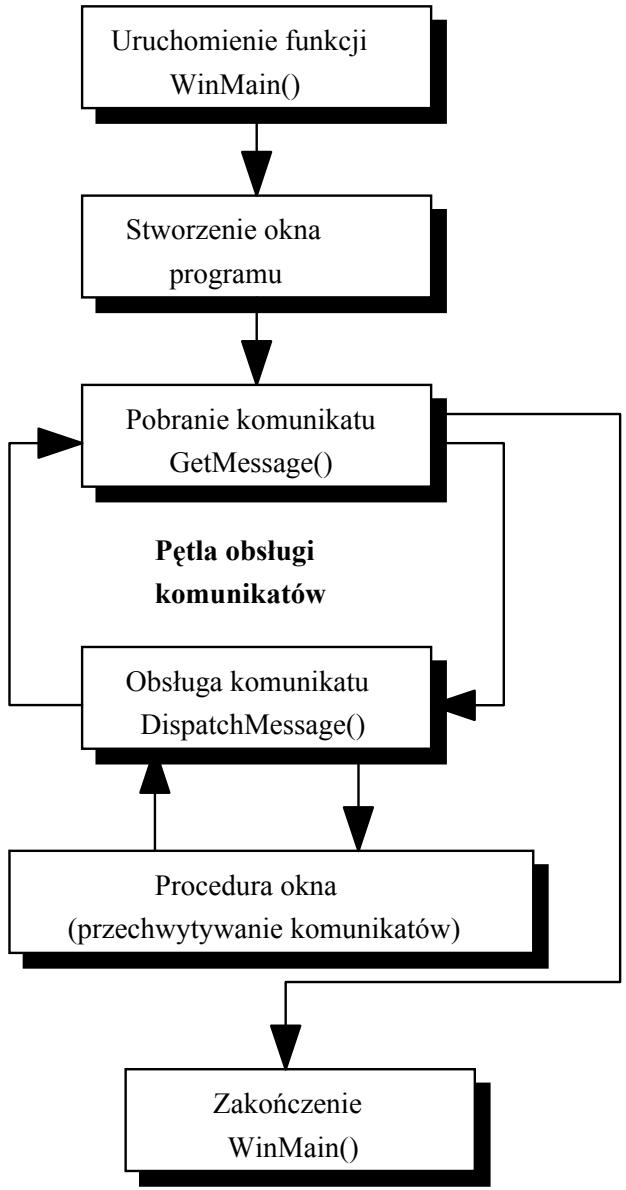
Pole nazwy nowego pliku tekstowego

# Programowanie Windows - okno przykładowej aplikacji



# Aplikacja Windows – fazy wykonywania programu

1. **Punkt startowy – funkcja WinMain()**
2. **Rejestrowanie klasy okna**
3. **Tworzenie okna**
4. **Wyświetlanie okna**
5. **Pętla komunikatów**
6. **Procedura okna**



# Aplikacja Windows – Punkt startowy – funkcja WinMain()

```
#define STRICT          // żądanie ścisłego przestrzegania
                        // zgodności typów w programie
#include <windows.h>// plik nagłówkowy funkcji API
                        // Windows

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
                                    // Prototyp głównej funkcji
                                    // obsługi głównego okna
    programu
// Główna funkcja programu:
int WINAPI WinMain
(   HINSTANCE hInstance,      // uchwyt instancji programu
    HINSTANCE hPrevInstance,// 0
    PSTR szCmdLine,           // linia komend
    int iCmdShow)            // początkowy stan programu
```

# Aplikacja Windows – rejestracja klasy okna

```
WNDCLASS wndclass;  // struktura do konstruowania klasy okna

// Tworzenie klasy głównego okna:
wndclass.style=CS_HREDRAW|CS_VREDRAW; // styl klasy okna
wndclass.lpfnWndProc=WndProc;     // nazwa głównej procedury
                                  // sterującej okna
wndclass.cbClsExtra=0;            // dodatkowy obszar pamięci
wndclass.cbWndExtra=0;            // dodatkowy obszar pamięci
wndclass.hInstance=hInstance;     // instancja programu
wndclass.hIcon=LoadIcon(NULL,IDI_APPLICATION);
                                  // ikona klasy okien
wndclass.hCursor=LoadCursor(NULL,IDC_ARROW);  // kształt kursora
wndclass.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH);
                                                // kolor tła
wndclass.lpszMenuName=MAKEINTRESOURCE(IDR_MENU1);
                                              // przyłączone MENU
wndclass.lpszClassName=szAppName;             // indywidualna nazwa
                                              // klasy okna

RegisterClass(&wndclass); // zarejestrowanie klasy
```

# Aplikacja Windows – tworzenie okna

```
hwnd=CreateWindow(
    szAppName,              // Nazwa klasy okna
    "Szablon1",             // Napis w  pasku tytułu
    WS_OVERLAPPEDWINDOW,    // Styl okna
    CW_USEDEFAULT,          // położenie okna x
    CW_USEDEFAULT,          // położenie okna y
    CW_USEDEFAULT,          // wysokość okna
    CW_USEDEFAULT,          // szerokość okna
    NULL,                   // uchwyt do okna rodzica
    NULL,                   // uchwyt do menu lub identy-
                            // fikator okna dziecka
    hInstance,              // uchwyt do instancji programu
    NULL                    // uchwyt do danych tworzących okno
              );
```

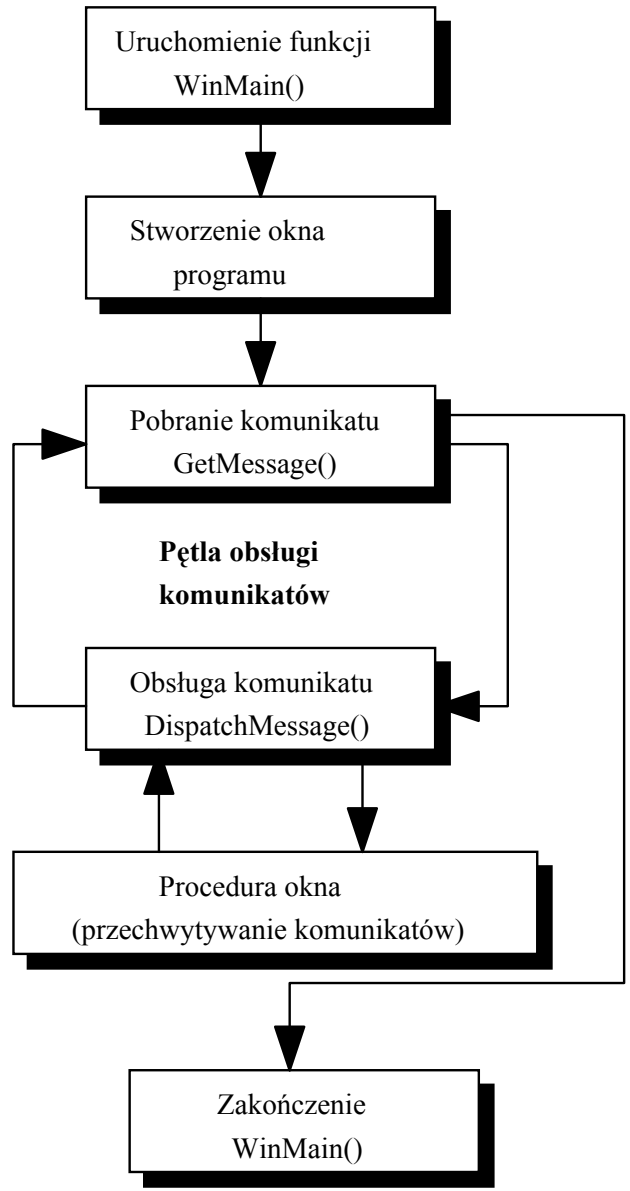
# Aplikacja Windows – wyświetlenie okna

```
ShowWindow(hwnd,iCmdShow);   // przygotuj okno do wyświetlenia

UpdateWindow(hwnd);      // wyślij pierwszy komunikat WM_PAINT
```

# Aplikacja Windows – pętla komunikatów

```
while(GetMessage(&msg,NULL,0,0))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
return msg.wParam;
```



# Aplikacja Windows – procedura okna

```
LRESULT CALLBACK WndProc(  WND hwnd,
                           UINT iMsg,
                           WPARAM wParam,
                           LPARAM lParam)
{   switch(iMsg)
    {
        case WM_CREATE:
              // obsłuż komunikat WM_CREATE i zwróć sterowanie
        case WM_SIZE:
              // obsłuż komunikat WM_SIZE i zwróć sterowanie
        case WM_COMMAND:
              // obsłuż komunikat WM_COMMAND i zwróć sterowanie
        case WM_DESTROY:
              // obsłuż komunikat WM_DESTROY i zwróć sterowanie
        case WM_PAINT:
              // obsłuż komunikat WM_PAINT i zwróć sterowanie
        case WM_CHAR:
              // obsłuż komunikat WM_CHAR i zwróć sterowanie
        default:
              return DefWindowProc(hwnd,iMsg,wParam,lParam);
    }
}
```

# Aplikacja Windows – rysowanie okna

```
case WM_PAINT: // Komunikat przychodzi w przypadku konieczności
               // "odmalowania" lub namalowania okna

               // Wymaż okno, uzyskaj kontekst urządzenia:
               HDCPaint=BeginPaint(hwnd,&PaintStruct);

               // Pobierz rozmiary obszaru roboczego okna
               GetClientRect(hwnd,&rect);

               // Umieść centralnie w oknie tekst:
               DrawText(HDCPaint,"Pozdrowienia",-1,&rect,
                                 DT_SINGLELINE|DT_CENTER|DT_VCENTER);

               // Zakończ rysowanie:
               EndPaint(hwnd, &PaintStruct);
               return 0;
```

# Aplikacja Windows – obsługa poleceń menu

```
case WM_COMMAND:      // Komunikat przychodzący po wybraniu
                      // opcji menu itp
              switch(LOWORD(wParam))
              {
                      // Wybrano opcję menu Plik->Koniec:
                      case ID_PLIK_KONIEC:
                              DestroyWindow(hwnd);
                              return 0;
              }
              return 0;
```

# Aplikacja Windows – zakończenie pracy programu

```
case WM_DESTROY:      // Komunikat przychodzący gdy okno
                      // ma być zniszczone
            PostQuitMessage(0);
            return 0;
```

# Aplikacja Windows - tworzenia okna dialogowego

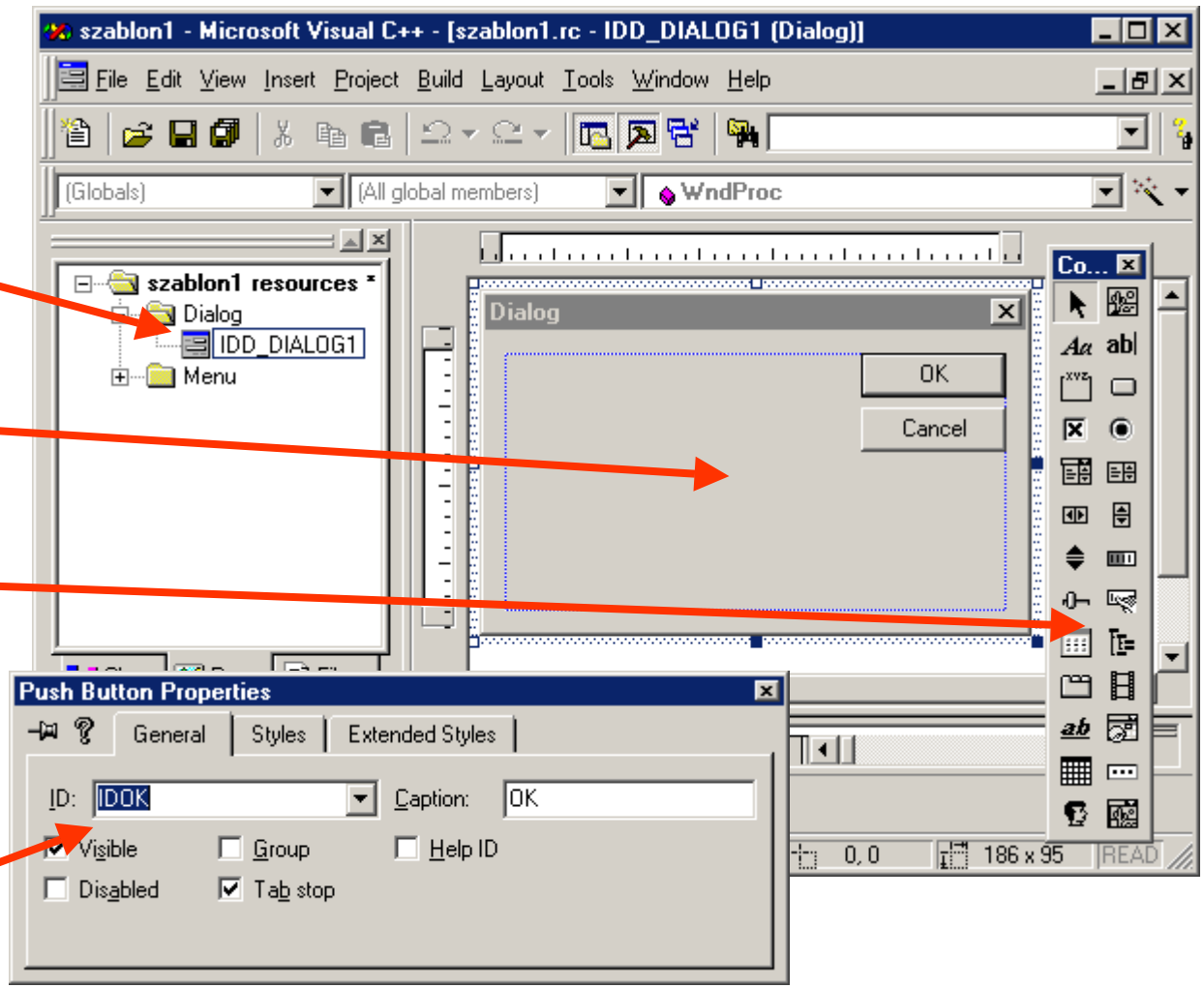
## 1. Insert -> Resource, wybrać dialog, new

Dołączony nowy zasób (IDD_DILALOG!)

Edycja okna dialogowego

Dostępne kontrolki okna dialogowego

Identyfikator przycisku OK

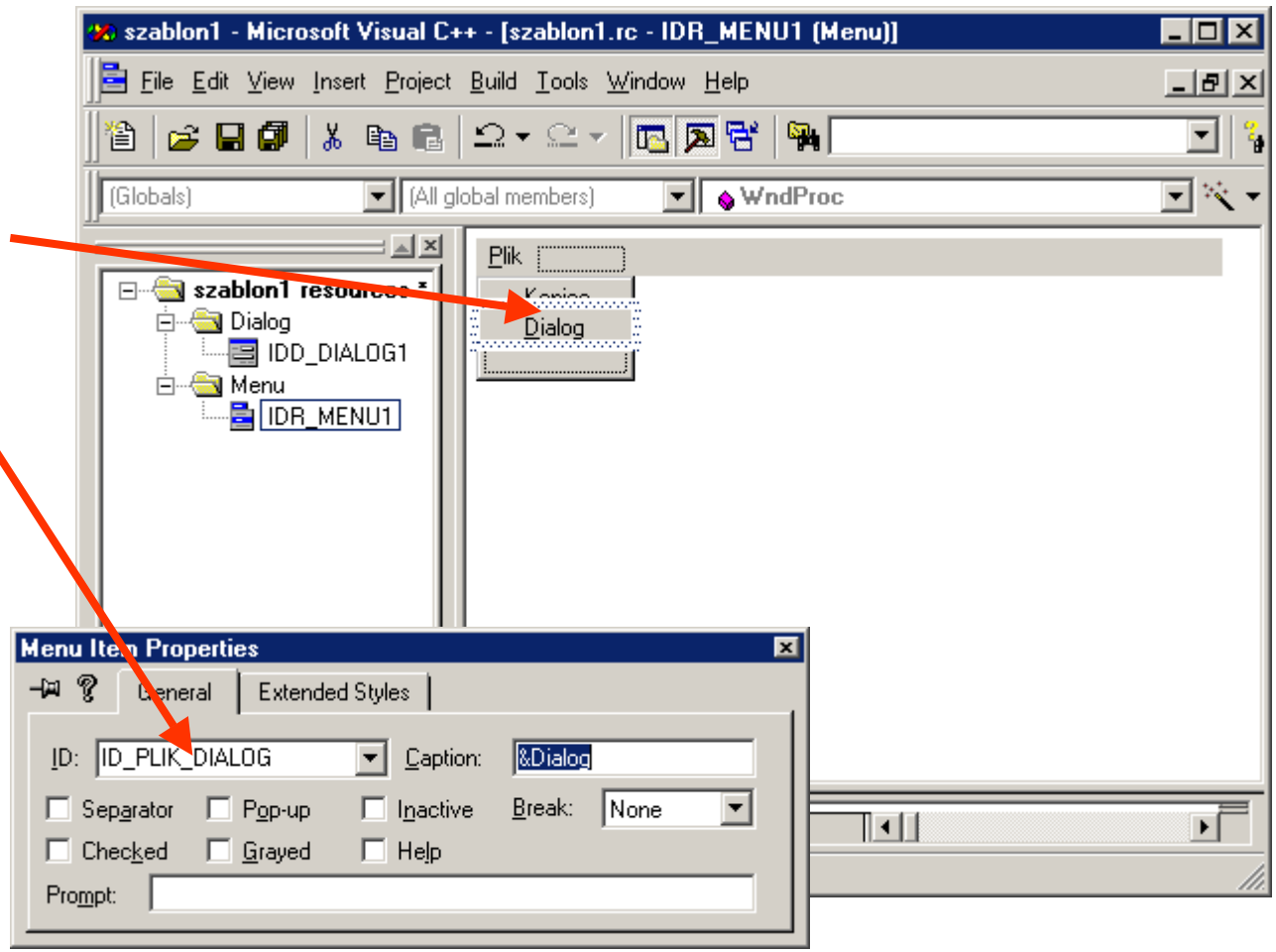


# Aplikacja Windows - tworzenia okna dialogowego

## 2. Uzupełnienie menu

**Dołączona nowa opcja:**

**Dialog (ID_PLIK_DIALOG)**



# Aplikacja Windows - tworzenia okna dialogowego

## 3. Uzupełnienie procedury okna

```
BOOL CALLBACK AboutDlgProc(HWND, UINT, WPARAM, LPARAM);
                                        // Prototyp funkcji obsługi
                                        // okna dialogowego
LRESULT CALLBACK WndProc
        (HWND hwnd, UINT iMsg, WPARAM wParam,LPARAM lParam)
{//...
static HINSTANCE hInstance; // Zmienna do zachowania instancji
switch(iMsg)
{   case WM_CREATE:              // Zachowanie instancji
       hInstance = ((LPCREATESTRUCT) lParam) -> hInstance;
       return 0;
    case WM_COMMAND:
       switch(LOWORD(wParam))
       {case ID_PLIK_KONIEC:DestroyWindow(hwnd); return 0;
        case ID_PLIK_DIALOG: // Wywołanie procedury okna
             DialogBox(hInstance,MAKEINTRESOURCE(IDD_DIALOG1),
                       hwnd, AboutDlgProc); return 0;
       }return 0;
//... }
```

# Aplikacja Windows - tworzenia okna dialogowego

## 4. Utworzenie procedury okna dialogowego

```
BOOL CALLBACK AboutDlgProc (HWND hDlg, UINT message,
                            WPARAM wParam, LPARAM lParam)
{
    switch(message)
    {
    case WM_INITDIALOG: // Zapewnienie wyświetlenia dialogu
        return TRUE;
    case WM_COMMAND:    // Obsługa kontrolek
        switch(LOWORD (wParam))
        {
        case IDOK:
        case IDCANCEL:
                EndDialog(hDlg,0);
                return TRUE;
        } break;
    }
    return FALSE;
}
```