# A Biblioteka WinBGIm

creative commons Jacek Tarasiuk

## A.1 Powstanie biblioteki WinBGI(m).

Na początku lat 90-tych XX wieku firma Borland udostępniła pierwszą prostą bibliotekę graficzną współpracującą z jej kompilatorami z serii Turbo. Na wiele lat biblioteka BGI (Borland Graphics Interface) stała się nieformalnym standardem tworzenia aplikacji graficznych. Tysiące ludzi na całym świecie stworzyło dziesiątki tysięcy aplikacji działających w oparciu o tą bibliotekę. Kiedy kompilatory DOS-owskie odeszły do lamusa pojawiła się potrzeba stworzenia biblioteki o zbliżonej funkcjonalności, dzięki której można by bez większych przeróbek przenieść istniejące oprogramowanie pod system Windows. Tak powstało WinBGI. Okazało się, że biblioteka jest na tyle łatwa i użyteczna, że bardzo szybko zaczęły pojawiać się wykorzystujące ją zupełnie nowe aplikacje. Pierwszą wersję tej biblioteki napisał Konstantin Knizhnik[1]. Następnie Michael Main wzbogacił ją o trzy funkcje w prosty sposób obsługujące myszkę, aż wreszcie Mark Richardson zniósł ograniczenie biblioteki polegające na obsłudze jedynie szesnastu kolorów. Obecnie biblioteka nazywa się WinBGIm (*m od mouse*) i można ją ściągnąć ze strony: `http://www.cs.colorado.edu/~main/cs1300/bgi/`

1 http://www.garret.ru/~knizhnik/

Tamże można znaleźć linki do stron z opisem biblioteki i zawartych w niej funkcji.

Poniżej zamieszczono opis wszystkich funkcji zawartych w bibliotece WinBGIm. Funkcje, które nie mają swoich odpowiedników w bibliotece Borlanda lub których nie zaimplementowano w bibliotece WinBGIm oznaczono specjalną informacją na marginesie. Część funkcji przeniesiono do biblioteki WinBGIm tylko w celu zachowania zgodności z biblioteką Borlanda. Ich używanie pod Windows nie ma za bardzo sensu. Funkcje takie opisano używając czcionki w szarym kolorze. Są to głównie funkcje dotyczące inicjalizacji i określania parametrów pracy trybów graficznych oraz funkcje dotyczące obsługi palety barw.

## *A.2 Funkcje trybu graficznego*

*Funkcje opisane poniżej wykorzystywane są w bibliotece BGI. W bibliotece WinBGIm zaimplementowano je w celu zachowania zgodności, ale w zasadzie są bezużyteczne. Zastąpiła je funkcja* **`initwindow`**, *której należy używać do inicjalizacji trybu graficznego.*

```
void initgraph(int far *graphdriver,
int far *graphmode, char far *pathtodriver);
```

Inicjalizuje system graficzny.

**`graphdriver`** – wartość 0 (DETECT) odpowiada autodetekcji
**`graphmode`** – jeżeli autodetekcja to zwraca mod grafiki
**`pathtodriver`** – ścieżka do własnych driverów (obecnie zupełnie nieużyteczne)

```
int graphresult(void);
```

Zwraca kod błędu ostatniej operacji graficznej. Możliwe wartości to:

**`grOk, grNoInitGraph, grNotDetected, grFileNotFound, grInvalidDriver, grNoLoadMem, grNoScanMem, grNoFloodMem, grFontNotFound, grNoFontMem, grInvalidMode, grError, grIOerror, grInvalidFont, grInvalidFontNum, grInvalidDeviceNum`**

```
char* grapherrormsg(int errorcode);
```

Zwraca tekst odpowiadający rodzajowi błędu opisanego wartością **errorcode**.

```
int getgraphmode(void);
```

Zwraca używany tryb graficzny.

Obecnie ważny jest właściwie tylko jeden **VGAHi** (640x480/16 kolorów). Czasami może się jeszcze przydać **IBM8514Lo** (640x480/256 kolorów).

```
void setgraphmode(int mode);
```

Ustawia tryb graficzny.

Użyteczne jedynie **VGAHi** i **IBM8514Lo**.

```
void getaspectratio(int far *xasp, int far *yasp);
```

Zwraca rozdzielczość poziomą **xasp** i pionową **yasp**. Ich stosunek daje *Aspect Ratio*.

```
void restorecrtmode(void);
```

Przywraca ustawienia trybu tekstowego. Wraz z **setgraphmode** i **getgraphmode** umożliwia łatwe przełączanie się pomiędzy trybem graficznym i tekstowym.

```
void detectgraph(int *graphdriver,
int *graphmode);
```

Funkcja zwraca informację o aktualnym trybie graficznym i używanym sterowniku.

```
char* getdrivername(void);
```

Zwraca nazwę używanego sterownika.

```
int getmaxmode(void);
```

Zwraca liczbę obsługiwanych trybów graficznych.

```
char* getmodename(int mode_number);
```

Zwraca nazwę trybu graficznego o numerze **mode_number**.

```
void getmoderange(int graphdriver, int *lomode,
int *himode);
```

Informuje o zakresie dostępnych trybów graficznych.

```
void graphdefaults(void);
```

Funkcja: ustawia obszar rysowania na całe okno, przesuwa pisak do pozycji (0,0), ustawia standardową paletę oraz kolory tła i pisaka, ustawia standardowy wzorzec wypełnienia oraz justowanie tekstu do lewej.

**nie zaimplementowana w WinBGIm**

```
int installuserdriver(char *name, int huge
(*detect)(void));
```

Instaluje sterownik użytkownika.

**nie zaimplementowana w WinBGIm**

```
int registerbgidriver(void (*driver)(void));
```

Rejestruje sterownik BGI (dołączony na etapie linkowania).

## A.3 Operacje na pikselach

```
int getpixel(int x, int y);
```

Zwraca wartość koloru w punkcie o współrzędnych **(x,y)**.

```
void putpixel(int x, int y,int pixelcolor);
```

Wstawia w położeniu (x,y) punkt o kolorze **pixelcolor**.

## A.4 Podstawowe operacje graficzne

```
int getx(void);
```

Zwraca aktualną współrzędną x pisaka.

```
int gety(void);
```

Zwraca aktualną współrzędną y pisaka.

```
void moveto(int x, int y);
```

Przesuwa pisak (nie rysując) do pozycji **(x,y)**.

```
void moverel(int dx, int dy);
```

Przesuwa pisak do pozycji odległej o **(dx,dy)** od aktualnej pozycji.

```
void lineto(int x, int y);
```

Rysuje linię prostą od aktualnej pozycji do pozycji **(x,y)**.

```
void linerel(int dx, int dy);
```

Rysuje linię prostą od aktualnej pozycji do pozycji odległej o **(dx,dy)** od aktualnej pozycji.

```
void line(int x0, int y0, int x1, int y1);
```

Rysuje linie prostą od punktu **(x0,y0)** do punktu **(x1,y1)**.

```
void rectangle(int left, int top, int right, int
```

**bottom);**

Rysuje prostokąt o przeciwległych wierzchołkach **(left,top)** i **(right,bottom)**.

### void circle(int x, int y, int radius);

Rysuje okrąg o środku w punkcie **(x,y)** i promieniu **radius**.

### void arc(int x, int y, int stangle, int endangle, int radius);

Rysuje wycinek łuku o środku w punkcie **(x,y)** i promieniu **radius**, **stangle** podaje kąt od jakiego rozpoczyna sie rysowanie łuku, **endangle** podaje kąt zakończenia rysowania łuku. Kąty podaje się w stopniach i liczy przeciwnie do ruchu wskazówek zegara. 0º oznacza godzinę 3, a 90º godzinę 12.

### void getarccoords(struct arccoordstype *arccoords);

Funkcja zwraca w postaci struktury:

```
struct arccoordstype {
   int x, y;
   int xstart, ystart, xend, yend;
};
```

informacje o ostatnio używanych parametrach polecenia **arc**.

### void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);

Rysuje wycinek łuku elipsy o środku w punkcie **(x,y)** i długości półosi **xradius** i **yradius**. **stangle** podaje kąt od jakiego rozpoczyna sie rysowanie łuku, **endangle** podaje kąt zakończenia rysowania łuku. Kąty podaje się w stopniach i liczy przeciwnie do ruchu wskazówek zegara. 0º oznacza godzinę 3, a 90º godzinę 12.

### void fillellipse(int x, int y, int xradius, int yradius);

Rysuje elipsę wypełnioną aktualnym kolorem wypełnienia lub wzorem wypełnienia. Środek elipsy znajduje się w punkcie **(x,y)** a długości półosi wynoszą **xradius** i **yradius**.

### void sector(int x, int y, int stangle, int endangle, int xradius, int yradius);

Rysuje wypełniony aktualnym kolorem lub wzorcem wypełnienia wycinek łuku elipsy o środku w punkcie **(x,y)** i długości półosi **xradius** i **yradius**. **stangle** podaje kąt od jakiego rozpoczyna sie rysowanie łuku, **endangle** podaje kąt zakończenia rysowania łuku. Kąty podaje się w stopniach i liczy przeciwnie do ruchu wskazówek zegara. 0º

oznacza godzinę 3, a 90º godzinę 12.

```
void drawpoly(int numpoints,
int far polypoints[]);
```

Rysuje łamaną otwartą, której wierzchołki podane są w tablicy **polypoints[]**.

```
void fillpoly(int numpoints, int *polypoints);
```

Rysuje łamaną tak jak funkcja **drawpoly** , a następnie wypełnia ją aktualnym kolorem wypełnienia lub wzorem wypełnienia.

```
void setlinestyle(int linestyle, unsigned upatter,
int thickness);
```

Ustawia styl rysowanych linii.

**linestyle** – styl linii (**SOLID_LINE , DOTTED_LINE , CENTER_LINE , DASHED_LINE , USERBIT_LINE**)
**upatter** – bitowy wzorzec linii, gdy **linestyle=USERBIT_LINE**
**thickness** – grubość linii w pikselach

```
void getlinesettings(struct linesettingstype far
*lineinfo);
```

Zwraca aktualne ustawienia stylu linii. Ustawienia przechowywane są w strukturze:

```
struct linesettingstype {
                int       linestyle;
                unsigned  upattern;
                int       thickness;   };
```

## A.5 Operacje tekstowe

```
void outtext(char far *textstring);
```

Wyświetla tekst **textstring** w aktualnej pozycji pisaka.

```
void outtextxy(int x, int y, char far
*textstring);
```

Wyświetla tekst **textstring** na pozycji **(x,y)**.

```
void settextjustify(int horiz, int vert);
```

Ustawia poziome i pionowe pozycjonowanie tekstu.

**horiz - LEFT_TEXT , CENTER_TEXT , RIGHT_TEXT**

**vert - BOTTOM_TEXT , CENTER_TEXT , TOP_TEXT**

### int textheight(char far *textstring);

Zwraca wysokość tekstu **textstring**.

### int textwidth(char far *textstring);

Zwraca szerokość tekstu **textstring**.

### void settextstyle(int font, int direction, int charsize);

Ustala parametry rysowanego tekstu.

**font** - **DEFAULT_FONT** (font bitmapowy używany standardowo 8x8 pikseli) , **TRIPLEX_FONT , SMALL_FONT , SANS_SERIF_FONT , GOTHIC_FONT**
**direction** - **HORIZ_DIR** (od lewej do prawej) , VERT_DIR (od dołu do góry)
**charsize** - rozmiar znaków w pikselach

Jeżeli wystąpi błąd w inicjalizacji fontu to **graphresult** przyjmuje wartość pomiędzy -8 a -14.

### void setusercharsize(int multx, int divx, int multy, int divy);

Funkcja określa sposób powiększania rozmiarów czcionki. Rozmiary wszystkich znaków będą przemnażane przez **multx** i **multy** oraz dzielone przez **divx** i **divy** (odpowiednio w kierunkach x i y). Dla przykładu **multx=3** i **divx=2** spowoduje poszerzenie znaków półtora raza. Ustawienia funkcji działają tylko wtedy, gdy **charsize** w funkcji **settextstyle** jest ustawiony na zero.

### void gettextsettings(struct textsettingstype far *textinfo);

Zwraca aktualne parametry rysowanego tekstu. Parametry przechowywane są w strukturze:

```
struct textsettingstype {
        int  font;
        int  direction;
        int  charsize;
        int  horiz;
        int  vert;    };
```

**nie zaimplementowana w WinBGIm**

### int installuserfont(char *name);

Instaluje czcionki zdefiniowane przez użytkownika.

**nie zaimplementowana w WinBGIm**

### int registerbgidriver(void (*driver)(void));

Rejestruje czcionkę dołączoną na etapie linkowania.

## A.6 Operacje dotyczące kolorów

*Kolory w bibliotece WinBGIm mogą być reprezentowane albo w postaci jednego ze standardowych kolorów odziedziczonych z biblioteki BGI* **( BLACK , BLUE GREEN , CYAN , RED , MAGENTA , BROWN , LIGHTGRAY , DARKGRAY , LIGHTBLUE , LIGHTGREEN , LIGHTCYAN , LIGHTRED , LIGHTMAGENTA , YELLOW , WHITE)** *albo w postaci RGB. Obie postacie przechowywane są w zmiennej typu int. Do obsługi kolorów w bibliotece używa się szeregu funkcji i makr. Standardowa biblioteka posługiwała się paletą 16 kolorów. W WinBGIm nadal możliwe jest korzystanie z palety. Jednak znacznie wygodniejsze wydaje się używanie kolorów RGB z pominięciem mechanizmu palety.*

**`void setcolor(int color);`**
Ustawia aktualny kolor pisaka na kolor oznaczony numerem **`color`** w palecie kolorów.

**`int getcolor(void);`**
Zwraca aktualny numer koloru pisaka.

**`void setbkcolor(int color);`**
Ustawia kolor tła.

**`int getbkcolor(void);`**
Zwraca kolor tła.

**`int getmaxcolor(void);`**
Zwraca maksymalnie dostępną liczbę kolorów.

tylko WinBGIm **`IS_BGI_COLOR(v)`**
Zwraca true jeśli kolor **`v`** jest kolorem standardowym biblioteki BGI.

tylko WinBGIm **`IS_RGB_COLOR(v)`**
Zwraca true jeśli kolor **`v`** jest kolorem RGB.

tylko WinBGIm **`int converttorgb(int color);`**
Konwertuje kolor standardowy BGI na kolor RGB.

tylko WinBGIm **`int COLOR(int r, int g, int b);`**

Tworzy kolor RGB o zadanych składowych.

tylko WinBGIm **`RED_VALUE(v)`**

Zwraca składową czerwoną koloru **`v`**.

tylko WinBGIm **`GREEN_VALUE(v)`**

Zwraca składową zieloną koloru **`v`**.

tylko WinBGIm **`BLUE_VALUE(v)`**

Zwraca składową niebieską koloru **`v`**.

**`void setpalette(int colornum, int color);`**

Przyporządkowuje miejscu w palecie **`colornum`** jeden z kolorów:

**`color`** – standardowy kolor BGI

**`void getpalette(struct palettetype *palette);`**

Zwraca strukturę:

```
struct palettetype {
        unsigned char size;
        signed char colors[MAXCOLORS + 1];
    };
```

w której **`size`** informuje o rozmiarze palety, a każdy element tablicy **`colors[]`** informuje o kolorze przechowywanym w danym miejscu w palecie. Wartości od 0 do 15 wskazują na standardowe kolory BGI. Wartość -1 informuje, że pozycji w palecie przypisano kolor RGB.

**`int getpalettesize(void);`**

Podaje liczbę kolorów jakie można umieścić w palecie dla danego trybu graficznego.

**`struct palettetype* getdefaultpalette(void);`**

Zwraca wskaźnik do palety ustawionej przez sterownik w trakcie inicjalizacji.

**`void setallpalette(struct palettetype *palette);`**

Wypełnia aktualną paletę kolorami umieszczonymi w strukturze **`palette`**. Funkcja obsługuje wyłącznie standardowe kolory BGI. Do ustawiania kolorów RGB należy używać **`setrgbpalette`**.

## A.7 Wypełnianie obszarów

**`void floodfill(int x, int y, int border);`**

Wypełnia obszar ograniczony kolorem border aktualnie ustawionym kolorem i wzorem wypełnienia zaczynając od punktu **`(x,y)`**.

**`void setfillstyle(int pattern, int color);`**

Ustawia aktualny wzorzec wypełniania i kolor wypełniania.

**`pattern - EMPTY_FILL, SOLID_FILL, LINE_FILL, LTSLASH_FILL, SLASH_FILL, BKSLASH_FILL, LTBKSLASH_FILL, HATCH_FILL, XHATCH_FILL, INTERLEAVE_FILL, WIDE_DOT_FILL, CLOSE_DOT_FILL, USER_FILL.`**

Uwaga: parametru **`USER_FILL`** nie używamy w WinBGIm. W tej bibliotece wystarczy zdefiniować własny wzorzec wypełnienia aby był on używany.

**`void getfillsettings(struct fillsettingstype far *fillinfo);`**

Zwraca aktualne parametry wypełnień. Parametry zapisywane są w strukturze:

```
struct fillsettingstype {
        int  pattern;
        int  color;      };
```

**`void setfillpattern(char far *upattern, int color);`**

Pozwala zdefiniować własny wzorzec wypełnień. **`upattern`** jest ośmioznakową tablicą typu char. Każdy bit tablicy we wzorcu odpowiada jednemu pikselowi. Jeśli wartość bitu wynosi 1 to punkt zostanie wypełniony kolorem **`color`**, jeśli 0 to będzie wypełniony kolorem tła.

**`void getfillpattern(char far *pattern);`**

Zwraca aktualnie używany wzorzec wypełnienia.

## A.8 Zaawansowane operacje graficzne

**`void bar(int left, int top, int right, int bottom);`**

Rysuje wypełniony prostokąt o przeciwległych wierzchołkach w punktach **`(left,top)`**

oraz **(right,bottom)**.

### void bar3d(int left, int top, int right, int bottom, int depth, int topflag);

Rysuje trójwymiarowy słupek histogramu o przeciwległych wierzchołkach w punktach **(left,top)** oraz **(right,bottom)**.

**depth** kontroluje "głębokość" słupka

**topflag** - **true**: rysuje słupek w całości, **false**: rysuje słupek bez górnej ścianki

### void pieslice(int x, int y, int stangle, int endangle, int radius);

Rysuje wypełniony wycinek koła o środku w punkcie **(x,y)** i promieniu **radius**. **stangle** podaje kąt od jakiego rozpoczyna sie rysowanie wycinka, **endangle** podaje kąt zakończenia rysowania łuku. Kąty podaje się w stopniach i liczy przeciwnie do ruchu wskazówek zegara. 0º oznacza godzinę 3, a 90º godzinę 12.

### void getimage(int left, int top, int right, int bottom, void far *bitmap);

Zapisuje fragment ekranu wyznaczony przez prostokąt o przeciwległych wierzchołkach w punktach **(left,top)** oraz **(right,bottom)** do pamięci w miejscu wyznaczonym przez wskaźnik ***bitmap**.

### void putimage(int x, int y, void far *bitmap, int op);

Wyświetla na ekranie bitmapę, na którą pokazuje wskaźnik ***bitmap**, w taki sposób, że lewy górny narożnik bitmapy będzie wyświetlony w punkcie **(x,y)**.

**op** - parametr określający sposób wyświetlania:

| | |
|---|---|
| **COPY_PUT** | - kopiuje bez zmian |
| **XOR_PUT** | - wykonuje operację XOR na bitach bitmapy i tła |
| **OR_PUT** | - wykonuje operację XOR na bitach bitmapy i tła |
| **AND_PUT** | - wykonuje operację XOR na bitach bitmapy i tła |
| **NOT_PUT** | - wykonuje operację XOR na bitach bitmapy i tła |

### unsigned imagesize(int left, int top, int right, int bottom);

Zwraca rozmiary bitmapy wyznaczonej przez prostokąt o przeciwległych wierzchołkach w punktach **(left,top)** oraz **(right,bottom)**. Przydatne przy rezerwowaniu pamięci do przechowywania bitmapy.

### void setwritemode(int mode);

Ustawia tryb rysowania na **COPY_PUT** lub **XOR_PUT**. Ustawienia wpływają jedynie na

działanie funkcji: **line**, **linerel**, **lineto**, **rectangle** i **drawpoly**.

## A.9 Operacje dotyczące urządzenia

**void cleardevice(void);**

Czyści ekran.

**int getmaxx(void);**

Zwraca szerokość ekranu.

**int getmaxy(void);**

Zwraca wysokość ekranu.

**void setviewport(int left, int top, int right, int bottom, int clip);**

Ustanawia prostokątny obszar zdefiniowany przez przeciwległe wierzchołki w punktach **(left,top)** oraz **(right,bottom)**, do którego odnosić się będą wszystkie instrukcje rysowania. Lewy górny narożnik obszaru będzie miał współrzędne (0,0) i względem tego punktu będą liczone wszystkie współrzędne.

**clip** - określa czy rysunki mają być obcinane do wyznaczonego obszaru (**true**), czy też mogą poza ten obszar wykraczać (**false**)

**void clearviewport(void);**

Czyści zdefiniowany ostatnio obszar rysowania.

**void setactivepage(int page);**

Uaktywnia stronę graficzną **page**[2].

tylko WinBGIm **int getactivepage(void);**

Zwraca numer aktywnej strony.

**void setvisualpage(int page);**

Wyświetla na ekranie stronę graficzną **page**.

tylko WinBGIm **int getvisualpage(void);**

Zwraca numer aktualnie widocznej strony.

**void getviewsettings(struct viewporttype *viewport);**

2 Strony mogą być numerowane od zera, ale twórcy biblioteki sugerują używanie stron o numerach 1 i 2.

Funkcja zwraca strukturę:

```
struct viewporttype {
   int left, top, right, bottom;
   int clip;
};
```

zawierającą ostatnie parametry wywołania funkcji **setviewport**.

## A.10 Funkcje obsługi strumienia.

tylko WinBGIm **ostringstream bgiout;**

Biblioteka WinBGIm definiuje globalny strumień zwany **bgiout**, z którego można korzystać w ten sam sposób jak ze strumienia **cout**.

Teksty wpuszczone do strumienia pojawią się na ekranie dopiero po wywołaniu jednej z dwóch funkcji **outstream** i **outstreamxy**.

tylko WinBGIm **void outstream(ostringstream& out=bgiout);**

Umieszcza na ekranie w aktualnej pozycji kursora, tekst wysłany wcześniej do strumienia.

tylko WinBGIm **void outstreamxy(int x, int y, ostringstream& out=bgiout);**

Umieszcza w pozycji o współrzędnych **(x,y)** tekst, wysłany wcześniej do strumienia.

## A.11 Funkcje obsługi myszy i klawiatury.

Wszystkie zdarzenia związane z myszą umieszczane są w kolejce i oczekują na obsłużenie. Rodzaj zdarzenia określa zmienna **kind**, która może przyjmować wartości:

**WM_MOUSEMOVE , WM_LBUTTONDBLCLK , WM_LBUTTONDOWN ,
WM_LBUTTONUP , WM_MBUTTONDBLCLK , WM_MBUTTONDOWN ,
WM_MBUTTONUP , WM_RBUTTONDBLCLK , WM_RBUTTONDOWN ,
WM_RBUTTONUP.**

tylko WinBGIm **void getmouseclick(int kind, int& x, int& y);**

Zwraca położenie **(x,y)** w którym miało miejsce zdarzenia opisane zmienną **kind**.

tylko WinBGIm **`void clearmouseclick(int kind);`**

Zwraca informację o rodzaju ostatniego zdarzenia.

tylko WinBGIm **`int mousex(void); int mousey(void);`**

Funkcje zwracają aktualne pozycje **`x`** i **`y`** kursora myszy.

tylko WinBGIm **`bool ismouseclick(int kind);`**

Funkcja zwraca wartość **`true`**, jeżeli w kolejce nieobsłużonych zdarzeń znajduje się zdarzenie zadanego typu **`kind`**.

tylko WinBGIm **`void setmousequeuestatus(int kind, bool status=true);`**

Funkcja ustawia kolejkowanie zdarzeń określonego typu. Jeżeli dla zdarzenia **`kind`** ustawiony zostanie status **`false`**, to pamiętane będzie tylko ostatnie zdarzenie danego typu. Jeśli ustawiona będzie wartość **`true`**, to kolejkowane będą wszystkie zdarzenia danego typu. Domyślnie ustawiona jest wartość **`false`** dla wszystkich zdarzeń.

tylko WinBGIm **`void registermousehandler(int kind, void h(int,int));`**

Funkcja rejestrująca własny handler obsługujący zdarzenie typu **`kind`**. Funkcja użytkownika powinna być typu **`void`** i przyjmować dwa argumenty typu **`int`**. Po zarejestrowaniu, każde zdarzenie typu **`kind`** zostanie automatycznie obsłużone przez utworzoną funkcję.

**`int getch(void);`**

Funkcja zwraca kod ASCII wciśniętego klawisza. Jeśli wciśnięto klawisz funkcyjny, najpierw zwracana jest wartość zero, a następnie kod klawisza funkcyjnego. Dla ułatwienia operowania klawiszami funkcyjnymi w bibliotece zdefiniowano poniższe stałe:

```
#define KEY_HOME       71
#define KEY_UP         72
#define KEY_PGUP       73
#define KEY_LEFT       75
#define KEY_CENTER     76
#define KEY_RIGHT      77
#define KEY_END        79
#define KEY_DOWN       80
#define KEY_PGDN       81
#define KEY_INSERT     82
#define KEY_DELETE     83
#define KEY_F1         59
#define KEY_F2         60
#define KEY_F3         61
#define KEY_F4         62
```

```
#define KEY_F5          63
#define KEY_F6          64
#define KEY_F7          65
#define KEY_F8          66
#define KEY_F9          67
```

**`int kbhit(void);`**

Funkcja zwraca wartość niezerową jeżeli w buforze klawiatury jakieś znaki oczekują na odczytanie. W przeciwnym wypadku zwracana jest wartość zero.

## A.12 Funkcje obsługi okien.

tylko WinBGIm

**`int initwindow(int width, int height, const char* title="Windows BGI", int left=0, int top=0, bool dbflag=false, bool closeflag=true);`**

Funkcja tworzy nowe okno graficzne zwracając jego identyfikator.

**`width`** – szerokość tworzonego okna

**`height`** – wysokość tworzonego okna

**`title`** – napis na belce tytułowej okna
(jeśli się wstawi pusty ciąg to okno zostanie wyświetlone bez belki tytułowej)

**`left , top`** – położenie lewego górnego narożnika okna

**`dbflag`** – ***true*** włącza podwójne buforowanie okna

**`closeflag`** – ***true*** zezwala na zamykanie okna ikonką [x]

zmiana w stosunku do BGI

**`void closegraph(int window=ALL_WINDOWS);`**

W standardowej bibliotece BGI funkcja kończy tryb graficzny, w WinBGIm zamyka okno o podanym identyfikatorze.

tylko WinBGIm

**`int getcurrentwindow(void);`**

Zwraca identyfikator okna, w którym wykonywane są operacje graficzne.

tylko WinBGIm

**`void setcurrentwindow(int window);`**

Ustawia okno o podanym identyfikatorze jako aktywne. Wszystkie operacje graficzne będą wykonywane w tym oknie.

tylko WinBGIm

**`int getmaxheight(void);`**

Zwraca maksymalną możliwą wysokość okna, które może być utworzone.

tylko WinBGIm **`int getmaxwidth(void);`**

Zwraca maksymalną możliwą szerokość okna, które może być utworzone.

tylko WinBGIm **`int getwindowheight(void);`**

Zwraca aktualną wysokość okna włączając w to ramki i pasek tytułowy.

tylko WinBGIm **`int getwindowwidth(void);`**

Zwraca aktualną szerokość okna włączając w to ramki.

## *A.13 Funkcje dodatkowe.*

tylko WinBGIm **`void delay(int millisec);`**

Wstrzymuje wykonanie programu na zadany okres czasu **`millisec`** wyrażony w milisekundach.

tylko WinBGIm **`int getdisplaycolor(int color);`**

Zwraca kolor jaki zostanie wyświetlony na urządzeniu, jeśli będziemy próbowali wyświetlić kolor **`color`**. Wartości te mogą się różnić w sytuacji, gdy urządzenie nie potrafi wyświetlić wszystkich możliwych kolorów. Wówczas zwrócona zostanie wartość koloru najbliższego żądanemu.

tylko WinBGIm
```
void printimage(const char* title=NULL,
double width_inches=7,double border_left_inches=0.
75, double border_top_inches=0.75,int left=0,
int right=0, int right=INT_MAX,
int bottom=INT_MAX);
```

Otwiera okno dialogowe drukarki, a następnie drukuje wybrany fragment okna.

**`title`** – nazwa zadania, która wyświetli się w menadżerze wydruku

**`width_inches`** – szerokość wydruku w calach

**`border_left_inches`** – lewy margines w calach

**`border_top_inches`** – górny margines w calach

**`left , right`** – lewy górny narożnik obszaru do wydruku

**`right , bottom`** – prawy dolny narożnik obszaru do wydruku

tylko WinBGIm **`int showerrorbox(const char *message);`**

Wyświetla okno z komunikatem **`message`** i oczekuje na wciśnięcie przycisku OK przez użytkownika.

tylko WinBGIm **`int swapbuffers(void);`**

Przełącza aktywną oraz widoczną stronę.

tylko WinBGIm **`void writeimagefile(const char* filename=NULL, int left=0, int top=0, int right=INT_MAX, int bottom=INT_MAX);`**

Zapisuje wybrany fragment okna do pliku w formacie BMP.

**`filename`** – nazwa pliku (jeśli NULL otwarte zostanie okno dialogowe z prośbą o podanie nazwy pliku)

**`left , right`** – lewy górny narożnik obszaru do zapisu

**`right , bottom`** – prawy dolny narożnik obszaru do zapisu

tylko WinBGIm **`void readimagefile(const char* title=NULL, int left=0, int right=0, int right=INT_MAX, int bottom=INT_MAX);`**

Funkcja wczytuje obrazek z pliku w formacie BMP, GIF, JPG, ICO, EMF lub WMF i wyświetla go w wybranym fragmencie aktywnego okna.

**`filename`** – nazwa pliku (jeśli NULL otwarte zostanie okno dialogowe z prośbą o podanie nazwy pliku)

**`left , right`** – lewy górny narożnik obszaru, w którym wyświetlony zostanie wczytany obrazek

**`right , bottom`** – prawy dolny narożnik obszaru, w którym wyświetlony zostanie wczytany obrazek

## *A.14 Kompilacja programów korzystających z biblioteki WinBGIm w środowisku DEV-C++ oraz wxDEV-C++.*

*Opis dotyczy środowiska Dev-C++ ver.4.9.9.2 i wxDec-C++ ver.6.10.2*

**1.** Skompilowaną bibliotekę "libbgi.a" umieścić w katalogu:

**`C:\Dev-Cpp\lib`**.

**2.** Plik nagłówkowy "graphics.h"[3] umieścić w katalogu:

3 Uwaga, niektórzy wykorzystują nagłówek „winbgim.h”, który jest identyczny z „graphics.h”, można więc powielić plik „graphics.h” zmieniając mu nazwę.

`C:\Dev-Cpp\include`.

**3.** Stworzyć nowy projekt jako "Console Application"[4].

**4.** Do projektu dodać pliki z programem.

**5.** W opcjach projektu, w zakładce "Parametry" w polu "Konsolidator" umieścić opcję dołączania bibliotek: "-lbgi -lgdi32 -lcomdlg32 -luuid -loleaut32 -lole32"[5].

**6.** Skompilować i uruchomić program.

**UWAGA**

Jeśli mamy problemy z kompilacją powodem może być niezbyt ścisłe przestrzeganie standardu C++, na które nowy kompilator Dev-C++ jest bardzo wyczulony. Zazwyczaj pomaga zadeklarowanie typu funkcji main jako innego niż void oraz wstawienie w opcjach projektu, w zakładce "Parametry" w polu "Kompilator C++" opcji "`-Wnodeprecated`".

## *A.15 Programy demonstracyjne*

Do skryptu dołączono dziesięć programów demonstrujących możliwości i wykorzystanie biblioteki WinBGIm. Są to kolejno:

1. **BGIdemo** – oryginalne demo (z 1993 roku) biblioteki BGI firmy Borland
2. **Inicjalizacja** – przykład inicjalizacji biblioteki oraz operacji na pikselach
3. **Linie** – rysowani linii, operowanie kolorem i wypełnieniami
4. **Text** – sposoby wypisywania tekstów
5. **Bufory** – przykład użycia dwóch stron graficznych z możliwością przełączania strony widocznej i strony aktywnej
6. **Funkcje zaawansowane** – wykorzystanie kopiowania blokowego fragmentów obrazu
7. **TrueColors** – demo pochodzące ze strony biblioteki WinBGIm pokazujące wykorzystania wszystkich 16mln kolorów

4 Jeśli chcemy używać tylko graficznego okna Windows bez okna tekstowego, wybieramy jako typ projektu „Win32 GUI”. Jeśli opcja ta nie jest dostępna w menu tworzenia projektów, możemy ją zawsze włączyć później we własnościach projektu.

5 Jeśli chcemy używać tylko graficznego okna Windows bez okna tekstowego dodajemy jeszcze: -mwindows.

8. **Pliki i drukarka** – wczytywanie i zapis grafiki oraz jej drukowanie na drukarce
9. **Obsługa myszy** – sposoby obsługi myszy
10. **Dwa okna** – jednoczesna praca z dwoma oknami graficznymi

## *A.16 Alfabetyczny spis funkcji*

```
BLUE_VALUE(v) (makro)
COLOR(int r,int g,int b)
GREEN_VALUE(v) (makro)
IS_BGI_COLOR(v) (makro)
IS_RGB_COLOR(v) (makro)
RED_VALUE(v) (makro)
arc (int x, int y, int stangle, int endangle, int radius);
bar (int left, int top, int right, int bottom);
bar3d (int left, int top, int right, int bottom, int depth, int
topflag);
circle (int x, int y, int radius);
cleardevice (void);
clearmouseclick(int kind);
clearviewport (void);
closegraph (int window);
converttorgb (int color);
delay (int millisec);
detectgraph (int *graphdriver, int *graphmode);
drawpoly (int numpoints, int *polypoints);
ellipse (int x, int y, int stangle, int endangle, int xradius, int
yradius);
fillellipse (int x, int y, int xradius, int yradius);
fillpoly (int numpoints, int *polypoints);
floodfill (int x, int y, int border);
getactivepage (void);
getarccoords (struct arccoordstype *arccoords);
getaspectratio (int *xasp, int *yasp);
getbkcolor (void);
```

```
getch (void);
getcolor (void);
getcurrentwindow (void);
getdisplaycolor (int color);
getdrivername (void);
getfillpattern (char *pattern);
getfillsettings (struct fillsettingstype *fillinfo);
getgraphmode (void);
getimage (int left, int top, int right, int bottom, void *bitmap);
getlinesettings (struct linesettingstype *lineinfo);
getmaxcolor (void);
getmaxheight (void);
getmaxmode (void);
getmaxwidth (void);
getmaxx (void);
getmaxy (void);
getmodename (int mode_number);
getmoderange (int graphdriver, int *lomode, int *himode);
getmouseclick(int kind, int& x, int& y);
getpalette (struct palettetype *palette);
getpalettesize (void);
getpixel (int x, int y);
gettextsettings (struct textsettingstype *texttypeinfo);
getviewsettings (struct viewporttype *viewport);
getvisualpage (void);
getwindowheight (void);
getwindowwidth (void);
getx (void);
gety (void);
graphdefaults (void);
grapherrormsg (int errorcode);
graphresult(void);
imagesize (int left, int top, int right, int bottom);
```

```
initgraph (int *graphdriver, int *graphmode, char *pathtodriver);

initwindow (int width, int height, const char* title, int left,
int top, bool dbflag, bool closeflag);

installuserdriver (char *name, int huge (*detect)(void));

installuserfont (char *name);

ismouseclick(int kind);

kbhit (void);

line (int x1, int y1, int x2, int y2);

linerel (int dx, int dy);

lineto (int x, int y);

mousex (void);

mousey (void);

moverel (int dx, int dy);

moveto (int x, int y);

outtext (char *textstring);

outtextxy (int x, int y, char *textstring);

palettetype* getdefaultpalette (void);

pieslice (int x, int y, int stangle, int endangle, int radius);

printimage (const char* title, double width_inches,double
border_left_inches, double border_top_inches,int left, int right,
int right, int bottom);

putimage (int left, int top, void *bitmap, int op);

putpixel (int x, int y, int color);

readimagefile (const char* filename,int left, int top, int right,
int bottom);

rectangle (int left, int top, int right, int bottom);

registerbgidriver (void (*driver)(void));

registerbgifont (void (*font)(void));

registermousehandler (int kind, void h(int, int));

restorecrtmode (void);

sector (int x, int y, int stangle, int endangle, int xradius, int
yradius);

setactivepage (int page);

setallpalette (struct palettetype *palette);

setaspectratio (int xasp, int yasp);
```

```
setbkcolor (int color);
setcolor (int color);
setcurrentwindow (int window);
setfillpattern (char *upattern, int color);
setfillstyle (int pattern, int color);
setgraphbufsize (unsigned bufsize);
setgraphmode (int mode);
setlinestyle (int linestyle, unsigned upattern, int thickness);
setmousequeuestatus(int kind, bool status);
setpalette (int colornum, int color);
setrgbpalette (int colornum, int red, int green, int blue);
settextjustify (int horiz, int vert);
settextstyle (int font, int direction, int charsize);
setusercharsize (int multx, int divx, int multy, int divy);
setviewport (int left, int top, int right, int bottom, int clip);
setvisualpage (int page);
setwritemode (int mode);
showerrorbox (const char *message);
swapbuffers (void);
textheight (char *textstring);
textwidth (char *textstring);
writeimagefile (const char* filename, int left, int top, int
right, int bottom);
```

Niniejszy tekst jest fragmentem skryptu do wykładu:

**Praktyczne wprowadzenie do grafiki komputerowej**
Skrypt ten w całości podlega licencji Creative Commons. Szczegółowy opis licencji znajduje się w przedmowie, dostępnej wraz z najnowszą wersją skryptu na stronie:
**http://novell.ftj.agh.edu.pl/~tarasiuk/dydaktyka/gfk/gfk.html**