Affichage dans une fenêtre graphique.

La bibliothèque **winbgi** contient des fonctions graphiques facilement utilisables en C++ sous CodeBlocks ou autre compilateur. Avant de pouvoir utiliser ces fonctions, commencez par **copier les fichiers** `graphics.h` et `winbgi.cpp` (qui contiennent le code des fonctions graphiques) depuis le dossier `Enseignement(V:)\winbgi` vers le dossier contenant votre programme. Ensuite, dans le menu **Projects** de CodeBlocks sélectionnez **Add files**, puis dans la fenêtre qui s'ouvre sélectionnez les deux fichiers `graphics.h` et `winbgi.cpp` et cliquez sur **Ouvrir**. Dans la fenêtre suivante, cliquez sur **OK**.
Dans le menu **Settings / Compiler,** onglet **Compiler settings**, sous **Other linker options**, tapez `-lgdi32` et cliquez sur **OK**.

Pour utiliser les fonctions graphiques, il faut ajouter dans votre programme l'inclusion :
#include "graphics.h"

Voici une liste (incomplète) des fonctions graphiques disponibles :

void opengraphsize(int larg, int haut)
ouvre une fenêtre graphique de largeur `larg` pixels et de hauteur `haut` pixels (p. ex. 600 par 450) ; à appeler en premier avant toute autre instruction du programme.

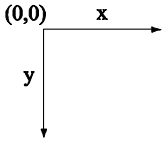
void closegraph()
ferme la fenêtre graphique ; à appeler avant de terminer le programme.

int getch()
attend que l'utilisateur tape sur une touche quelconque avant de poursuivre l'exécution du programme (renvoie aussi le code ASCII de la touche enfoncée, mais il n'est pas nécessaire de récupérer cette valeur).
ATTENTION : la fenêtre graphique doit être active lorsque la touche est enfoncée.

void plot(int x, int y)
affiche le pixel (`x`,`y`).

void line(int x1, int y1, int x2, int y2)
trace un segment du point (`x1`,`y1`) au point (`x2`,`y2`).

ATTENTION : l'origine du repère est le coin supérieur gauche de la fenêtre.



void lineto(int x, int y)
trace un segment depuis la position actuelle du curseur graphique jusqu'au point (`x`,`y`).

void moveto(int x, int y)
déplace le curseur graphique en (`x`,`y`) sans rien dessiner (les autres fonctions graphiques déplacent aussi le curseur graphique après avoir dessiné).

void rectangle(int x1, int y1, int x2, int y2)
trace le rectangle de coin supérieur gauche (`x1`,`y1`) et de coin inférieur droit (`x2`,`y2`).
void bar(int x1, int y1, int x2, int y2)
même fonction que la précédente mais remplit le rectangle tracé.

void circle(int x, int y, int rayon)
trace le cercle de centre (`x`,`y`) et de rayon `rayon`.

void drawpoly(int n, int tableauPoints[])
trace une ligne polygonale de n sommets. Les coordonnées des sommets sont stockées dans le tableau `tableauPoints` qui contient 2`n` entiers. tableauPoints[0] et tableauPoints[1] sont les coordonnées (x,y) du premier sommet, …
ATTENTION : pour tracer un polygone fermé le premier sommet doit être répété à la fin du tableau (pour un triangle on aura donc n=4).

void fillpoly(int n, int tableauPoints[])
même fonction que **drawpoly** mais remplit le polygone tracé.

void setcolor(int codeCouleur)
fixe la couleur du pinceau qui est utilisée pour les tracés. 16 constantes entières qui correspondent à des codes de couleurs sont définies dans `graphics.h`. Il s'agit de : BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN, LIGHTGRAY, DARKGRAY, LIGHTBLUE, LIGHTGREEN, LIGHTCYAN, LIGHTRED, LIGHTMAGENTA, YELLOW, WHITE.

void setbkcolor(int codecouleur)
fixe la couleur de fond, utilisée notamment lors des effacements d'écran.

void cleardevice()
efface la fenêtre graphique.

Exemple : le programme suivant affiche une droite bleue sur fond rouge

```
#include "graphics.h"

int main()
{
  opengraphsize(1800,900);
  setbkcolor(RED);
  cleardevice();
  setcolor(BLUE);
  line(100,100,500,500);
  getch();
  closegraph();
}
```