研究ノート

C言語学習初心者のための簡易グラフィックス環境

磯　貝　芳　徳
日本福祉大学　情報社会科学部

A Simplified Graphics Environment for C Language Beginners

Yoshinori Isogai
Faculty of Social and Information Sciences, Nihon Fukushi University

Keywords: プログラミング言語教育，グラフィックス，C 言語，X-Womdow system

1．はじめに

プログラミング言語を学ぶ初心者にとって，簡易なグラフィックス環境の有無は学習効果の点で大きな差となる．プログラムの出力に数値や文字に加えて彩り豊かな画像が可能になれば，学習者は楽しみながらプログラミングの学習に取り組むことができよう．加えて画像出力ではプログラムの誤りが一目で分かるため，初心者であっても debugging の道筋を容易に発見できる． さらに処理の順番すなわち algorithm の重要性を容易に認識できるなど, プログラミング学習に有益な特色を有する．これらの利点は，非常に簡易なグラフィックス環境を提供する BASIC による教育体験を有する人々の等しく認めるところであろう．

C言語によるプログラミング教育はパーソナルコンピュータ（ＰＣ）あるいはワークステーション（ＷＳ）上の UNIX あるいは Linux の OS 環境で行われることが多い．しかもこれらの OS ではかつてのコンソールのみによる操作はすっかり影を潜め，X-Window System を GUI 環境として用いることが一般化している．X-Window System により機器の操作が極めて簡便になった一方で，ディスプレイに画像を出力しようとすると，プログラマは X-Window System のグラフィックス環境について通暁することが求められる．すなわち，このような環境でC言語を用いて画像出力を得るには，まず X-Window System の基本的な考え方とその膨大な関数群を学ぶ必要がある．加えてこれらを理解し使いこなすには，C言語学習上比較的高度とされる構造体，共用体，ポインタ，クラス等の概念を理解する必要がある．これらの修得はプログラミング言語の初学者には荷が重過ぎることは言うまでもない．このような高度な知識を有さずとも簡単にグラフィックス出力の得られる環境が求められる．

ここで簡易グラフィックス環境とは，C言語によるプログラミング学習を X-Window system が稼動している環境下で行う初学者を対象とし，上記のような X-Window system に関わる知識やC言語の高度な知識を前提とせず，簡単に画像出力が得られる環境を意味する．我々はこのようなグラフィックス環境を実現する system を構築したので報告する．学習者がこの環境を利用するには，我々が提供する include file を自己のプ

ログラムに include するだけでよい．

次章ではこの簡易グラフィックス環境構築にあたっての基本的な考え方と，この環境により利用者は何ができるかについて述べる．第３章では当環境の利用方法を述べ，いわばマニュアルを提供する．Appendix A には当簡易グラフィックス環境を用いたプログラム例を，Appendix B には当グラフィックス環境を実現するための include file のリストを掲げる．

2. 簡易グラフィックス環境の構築

簡易グラフィックス環境の構築にあたり次の４点に留意した．

1．利用者はC言語の構造体，共用体，ポインタおよびクラスといった概念を知らないことを前提とする．
2．利用者は X-Window system の display, window, graphics context および attribute [1] といった概念を知らないことを前提とする．
3．当グラフィックス環境が提供する描画要素は基本的なものに限定する．
4．実行時の効率を多少犠牲にしても，利用者の使い易さと debugging の容易さを優先する．
5．当グラフィックス環境は include file の形で提供される．これはコンパイル時の効率を多少犠牲にしても，初学者が理解しやすいことと，いつでもソースファイルを見たり，修正を加えることを可能にすることを目指したものである．

これらの点に留意して構築した include file のリストを Appendix B に示す．include file は大きく２つの部分からなる．第１の部分は描画用の window の生成，描画に関わる各種構造体の宣言文および描画のための各種関数の宣言文からなる．特に構造体の宣言文を外部変数として宣言することにより次のような利点が生まれた．本環境の主要部を構成する各関数内で，これら構造体を利用者のプログラムを経由することなく随意に用いることができ，利用者はこれらの構造体について何も知らなくてもよい環境を構築することが可能となった．

第２の部分は描画に関わる各種関数群である．この関数群は次のようなグループから構成されている．

1．描画用 window の生成，クリア処理，クローズ処理等に関わる関数
2．使用可能な色の設定に関わる関数
3．各種描画要素（点，線分，円，長方形および文字列）を描画する関数
4．描画用 window の印刷とファイルへの格納を行う関数

X-Window system を用いてグラフィックス出力を行う際，最も面倒な window の生成に関わる部分は全て関数 YOpenWin（Appendix B のリスト参照）で行う．利用者は描画用 window の幅と高さを引数を介してこの関数に渡すことで window の初期化を済ませることができる．加えて YOpenWin では本環境で色を簡便に利用するための準備作業を行う．これにより利用者は X-Window system における color map や pixel 値 [1] についての知識を一切必要とせず，色数は限定されるものの，各種描画要素の色を極めて簡便に制御することが可能になる（詳しくは Appendix B を参照）．

本環境で提供する描画要素は点，線分，円，長方形および文字列の５種類である．これらの要素を描画する際の座標系として本環境では次のような座標系を採用した．描画用 window（YOpenWin で初期化する）の左上隅を原点とし，右向きにX軸，下向きにY軸を採る．座標系の単位は pixel とする．各描画要素に対応する関数は全て色を表す引数が用意されており，各描画要素毎に色指定を行うことができる．また，各関数内では描画を終えたあと flush 処理（buffer 内の情報を display に吐き出す処理）を行うようにした．これは利用者に flush についての知識を要求しないで済ませることを目的としている．同時にこれにより描画 window で時間とともに生起する事象と利用者のプログラムの流れが一致し，初心者にとってプログラムの debugging が容易になるという利点が生まれた．描画要素毎に flush することによる実行効率の低下よりもこれらの利点を優先すべきであるとの判断に基づくものである．

描画 window 上の画像を印刷したり，file に出力するための関数も用意されている．印刷や file への出力は X-Window dump 機能 [1] をシステムコールする方法で実現した．従って出力 file の書式は xbm となる．

本簡易グラフィックス環境の詳細については Appendix B に掲げる include file のリストを参照されたい．

3. 簡易グラフィックス環境の利用法

最初に当グラフィックス環境の利用例を示して当環境の利用法の概要を述べ，ついで当環境で採用した座標系，色の扱い方，当環境が提供する個々の関数の具体的な説

明および compile 時の留意点を述べる．

Appendix A は当グラフィックス環境を用いて簡単な描画を行うプログラム例を示す．このプログラムを用いて描かれた画像を図 1 に示す．ただし図 1 に示す画像は白黒の単色印刷のため，実際の画面上の多彩な色が表現されていない点に留意されたい．

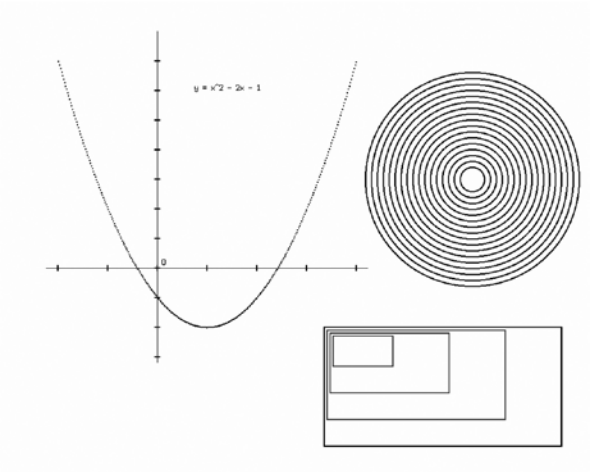

図 1 描画の出力例（Appendix A のプログラムによる）

Appendix A の第 1 行目の include 文では当環境が提供する include file（参照 Appendix B）を include している．当環境で用いる各種変数や構造体等の宣言文および描画のための関数群はすべてこの include file で提供されるため，この include 文は必須である．第 12 行目では描画用 window の幅と高さを引数に与えて YOpenWin 関数をコールし，描画用 window を開く．それ以降の行では各種の描画要素（点，線分，円，長方形および文字列）を描く関数をコールして描画を行う（リスト中の注釈文を参照）．第 55 行では描画 window の印刷あるいは file への格納を選択し実行する関数 PrintOrFile をコールする．第 65 行では描画 window のクローズ処理を行う関数 YCloseWin をコールし，プログラムを終了する．

以下では当環境で用いる座標系，色の扱い方，個々の関数の説明および compile 時の留意点を記す．いわば当簡易グラフィック環境を利用する上でのマニュアルを提供することを意図している．

① 座標系

YOpenWin 関数（下記参照）で生成した window の左上隅の点を原点とし，右方にＸ軸を，下方にＹ軸を設定する．座標系の単位は pixel とする．

② 描画要素の色

X-Window system で色を扱うには color map の概念を理解し，color map の pixel 値を扱う必要がある．当環境では扱える色数を１１種類に限定し，これらの色名（例えば red とか blue）のみで描画要素の色を指定できるようにした．色名は include file 内で unsigned long 型の変数として宣言され，対応する color map の pixel 値が格納されている．利用者は color map については一切知る必要がなく，色名のみで描画要素の色を制御できる．当環境で利用可能な色は次のとおりである．

```
black,white,red,blue,green,orange,yellow,
violet,darkgreen,purple,skyblue
```

③ 関数群

〈描画用 window の操作に関わる関数〉

１）`YOpenWin ( int w, int h)`

[機能] 描画用 window を生成する．

[引数] w：描画用 window の幅（pixel 値）

h：描画用 window の高さ（pixel 値）

２）`YClearWin ( )`

[機能] 描画用 window 内を背景色（白）で塗りつぶし，window 内をクリアする．

３）`YCloseWin ( )`

[機能] 描画用 window を閉じ，画面より消滅させる．

〈描画要素を描く関数〉

１）`YText (int x, int y, char str[], int n, unsigned long c)`

[機能] 文字列を表示する．

[引数] x：文字列の最初の文字の左下隅の x 座標

y：文字列の最初の文字の左下隅の y 座標

str[ ]：表示する文字列

n：表示する文字列の文字数

c：文字列の色

２）`YLine (int x1, int y1, int x2, int y2, unsigned long c, unsigned int lw, int ls)`

[機能] 点(x1, y1)と点(x2, y2)を結ぶ線分を描く．

[引数] x1, y1, x2 および y2：説明省略

c：線分の色

lw：線分の線幅（線の太さ）

ls：線の種類を指定．実線なら Solid，破線なら Dash を指定する．

3）YCircle (int x, int y, int r, unsigned long c, int fe)
[機能] 点 (x, y) を中心とする半径 r の円を描く.
[引数] x, y および r：説明省略
c：色を指定
fe：円の内部を塗りつぶすか (FILL)，塗りつぶさず線画にするか (EMPT) を指定

4）YRectangle (int x, int y, int w,int h, unsigned long c, int fe)
[機能] 点 (x, y) を左上の頂点とし，幅 w，高さ h の長方形を描く.
[引数] x, y, w および h：説明省略
c：色を指定
fe：長方形の内部を塗りつぶすか (FILL)，塗りつぶさず線画にするか (EMPT) を指定

5）YPoint (int x, int y, unsigned long c)
[機能] 点 (x, y) に点を描く.
[引数] x および y：説明省略
c：点の色を指定

〈描画 window の印刷，file への格納に関わる関数〉

1）PrintOrFile ( )
[機能] プログラムの実行命令を入力した window（通常は console window）に，描画 window の印刷かファイルへの格納かの選択メニューを表示する．印刷を選択するとカーソルが十文字形に変わる．このカーソルを描画 window に運び，マウスの左ボタンをクリックするとこの winodow の内容がプリンタに印刷される．file への格納を選択すると file 名の入力を促すメッセージが出るので file 名を入力し，以降は印刷の場合と同様にカーソルで描画 window を選択する.

④ compile 時の留意点

compile に先立ち，利用者作成のプログラムに当簡易グラフィックス環境が提供する include file が include されていることを確認する.

compile にあたってはいくつかの option parameter を付加する必要がある．下に compile のためのコマンド例を示すが，system によって path 名が異なる場合があるので system administrator に相談するとよい.

〈compile コマンドの例〉

```
% cc ソースファイル名 -L/usr/X11R6/lib -I/usr/X11R6/include -lX11 -lnsl -lsocket
```

上記の -L/usr/X11R6/lib では X-Window system の library 群の置かれている path 名を指定する．-I/usr/X11R6/include では X-Window の include file 群の置かれている path 名を指定する．-lX11，-lnsl および -lsocket では link すべき library を指定する.

4. まとめ

C 言語を学ぶ初心者用の簡易グラフィックス環境を構築し，その内容について報告した．この環境の利用者は X-Window system に関する諸知識および C 言語の高度な知識（構造体，クラス，ポインタ等）を有さずとも，簡単な関数呼び出しのみで window 上にグラフィックス出力を行うことができる.

当グラフィックス環境はプログラミング言語の初学者を対象にし，プログラミング教育に画像出力を加えることにより教育効果の向上を目指したものである．実際に日本福祉大学情報社会科学部の初心者用の C 言語教育科目および一部の卒業研究において，1996 年より数年間にわたって使用され，しかるべき成果をあげてきた．ただ教育上の効果等についての科学的な評価作業は行われていない．今後このような評価の作業と評価に基づく system の改良が課題として残されている.

付記

当システムの移植，利用については一切の制限を設けない．自由に移植，利用されたい．またここで示した include file の電子媒体での提供を行っている．希望される方は筆者へ連絡されたい.

参考文献

1）木下凌一，林秀幸：X-Window Ver. II プログラミング【第 2 版】，日刊工業新聞社，(1993)

Appendix A

簡易グラフィックス環境を用いたプログラム例を下に示す．図１はこのプログラムを用いて描かれた．

```
#include </home/isogai/C/isogai.h>
#include <stdio.h>
main() {
    int   Wwidth  = 1000;
    int   Wheight = 800;
    int   i, x, y, r;
    float xx, yy;
/* 描画用ウィンドウの幅と高さを与えてウィンドウを開く */
   YOpenWin (Wwidth, Wheight);
 /* 点を描く関数を用いて２次関数のグラフを描く　*/
    for(xx=-2.0; xx <= 4.0; xx += 0.02) {
      yy = xx * xx - 2 * xx - 1;
      x = (int) (84.0 * xx) + 268;
      y = 450 - (int) (50.0 * yy);
      YPoint(x, y, red);
     }
/* 線分を描く関数を用いてＸ軸とＹ軸および目盛りを描く */
    YLine ( 80, 450, 624, 450, black, 1, Solid);
    YLine (268,  50, 268, 610, black, 1, Solid);
    for(xx=-2.0; xx <= 4.0; xx += 1.0) {
      x = (int) (84.0 * xx) + 268;
      YLine ( x, 445, x, 455, black, 1, Solid);
    }
    for(yy=7.0; yy >= -3.0; yy -= 1.0) {
      y = 450 - (int) (50.0 * yy);
      YLine ( 263, y, 273, y, black, 1, Solid);
   }
/*   グラフの原点とタイトルを文字列を描く関数を用いて描く */
    YText ( 275, 445, "O", 1, black);
    YText( 330, 150, "y = x^2 - 2x - 1", 16, black);
/*  同心円を ,円を描く関数を用いて描く　　*/
    x = 800;
    y = 300;
    for( r=20; r<=180; r += 10) {
      YCircle ( x, y, r, blue, EMPT);
   }
/* ４つの長方形を描く　　*/
  YRectangle (550, 550, 400, 200, darkgreen, EMPT);
  YRectangle (555, 555, 300, 150, darkgreen, EMPT);
  YRectangle (560, 560, 200, 100, darkgreen, EMPT);
  YRectangle (565, 565, 100,  50, darkgreen, EMPT);
/* 描画画面の印刷，ファイルへの出力問い合わせ */
    PrintOrFile();
/* 描画画面のクリア．　　*/
/* 複数の画像を描いては印刷するといった場合に利用 */
    YClearWin();
/* 画面がクリアされたことを見てもらうため５秒間スリープ　*/
    sleep(5);
/* 描画用ウィンドウのクローズ処理　　*/
    YCloseWin();
 }
```

Appendix B

簡易グラフィックス環境を実現するための include file のリストを下に示す．

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <string.h>
/* 円や長方形の内部を塗りつぶす */
#define FILL    1
/* 円や長方形の内部を塗りづぶさない */
#define EMPT    0
/* ------------------------------------ */
/*           関数の宣言                 */
/* ------------------------------------ */
unsigned long YColor();
void YMyColor();
void YOpenWin(int, int);
void YCloseWin();
void YClearWin();
void YForegroundColor(unsigned long);
void YLine(int, int, int, int, unsigned long,
unsigned int, int);
void YRectangle(int, int, int, int, unsigned
long, int);
void YCircle(int, int, int, unsigned long,
int);
void YText(int, int, char [], int, unsigned
long);
void YPoint(int, int, unsigned long);
void YFlush();
void PrintOrFile();
/* 描画用ディスプレイのディスプレイ変数 */
Display *YDisp;
/* 描画用ウインドウのウインドウ変数　　*/
Window Ywin;
/* グラフィックスコンテキスト　　*/
GC Ygc;
/*     使用可能な色     */
unsigned long black, white, red, blue,
green, orange, yellow, violet, darkgreen,
purple, skyblue;
XSetWindowAttributes Yatt;
```

```
XEvent    e;
XSetWindowAttributes at;
int Solid = LineSolid;      /* 線種：実線 */
int Dash = LineOnOffDash;   /* 線種：破線 */

/*   初期化 (トップウインドウとその下の描画用のウインドウを用意)*/
void YOpenWin(int w, int h) {
  Font f;
  Window win;
  YDisp = XOpenDisplay(NULL);
/* トップウインドウ */
  win = XCreateSimpleWindow(YDisp,
  DefaultRootWindow (YDisp), 0, 0, w, h, 2,
  black, white);
/* 描画用ウインドウ */
  Ywin = XCreateSimpleWindow(YDisp, win,
  0, 0, w, h, 0,BlackPixel(YDisp, 0),
  WhitePixel(YDisp, 0));
/* 他のウインドウに隠された部分を      */
/* 再描画するための設定                 */
  at.backing_store = Always;
  XChangeWindowAttributes (YDisp, Ywin,
  CWBackingStore, &at);
  XChangeWindowAttributes (YDisp, win,
  CWBackingStore, &at);
  XSelectInput (YDisp, Ywin, ExposureMask);
  XSelectInput (YDisp, win, ExposureMask);
/* ウインドウの表示名 */
  XStoreName (YDisp, win, "My Graphics");
/* アイコンの表示名    */
  XSetIconName (YDisp, win, "My Graphics");
/* トップウインドウと描画用ウインドウのマッピング */
  XMapWindow (YDisp, win);
  XMapWindow (YDisp, Ywin);
  Ygc = XCreateGC (YDisp, Ywin, 0, 0);
/* テキストのフォントの設定    */
  f = XLoadFont (YDisp, "-adobe-courier-
  medium-r-normal-17-120-100-100-m-100-
  iso8859-1");
  XSetFont (YDisp, Ygc, f);
/* 使用可能色を設定する関数 YMyColor の呼び出し */
  YMyColor();
/* フォアグラウンド色の設定（実際は無意味）*/
  XSetForeground (YDisp, Ygc, black);
  XFlush(YDisp);
}
/*   描画用ウインドウのクローズ処理   */
void YCloseWin() {
  XDestroyWindow(YDisp, Ywin);
}
/*   描画用ウインドウのクリア処理      */
void YClearWin() {
  XSetWindowBackground (YDisp, Ywin, white);
  XClearWindow (YDisp, Ywin);
  XFlush(YDisp);
}
/*   バッファ内データのフラッシュ処理   */
void YFlush() {
  XFlush( YDisp );
}

/*   点の描画                                */
/*      x：点の x 座標                       */
/*      y：点の y 座標                       */
/*      c：点の色                            */
void YPoint(int x, int y, unsigned long c) {
  YForegroundColor ( c );
  XDrawPoint (YDisp, Ywin, Ygc, x, y);
  XFlush(YDisp);
}
/*   テキストの出力                          */
/*   x：テキストの最初の文字の左下のｘ座標    */
/*   y：テキストの最初の文字の左下のｙ座標    */
/*   str[ ]：テキストの内容の入っている string  */
/*   c：テキストの色                         */
void YText(int x, int y, char str[], int n,
unsigned long c) {
  YForegroundColor ( c );
  XDrawString (YDisp, Ywin, Ygc, x, y, str, n);
  XFlush(YDisp);
}

/*   線分の描画 */
/*      x1：線分の始点のｘ座標               */
/*      y1：線分の始点のｙ座標               */
/*      x2：線分の終点のｘ座標               */
/*      y2：線分の終点のｙ座標               */
/*       c：線分の色                         */
/*      lw：線幅（線の太さ）                 */
/*      ls：線種（Solid：実線 Dash：破線）   */
void YLine(int x1, int y1, int x2, int
y2, unsigned long c, unsigned int lw, int
ls) {
  YForegroundColor ( c );
  XSetLineAttributes (YDisp, Ygc, lw, ls,
  CapButt, JoinMiter);
  XDrawLine (YDisp, Ywin, Ygc, x1, y1, x2, y2);
  XFlush(YDisp);
}

/*   円の描画 */
/*      x：中心のｘ座標                      */
/*      y：中心のｙ座標                      */
/*      r： 半径                             */
/*      c：色                                */
```

```
/*    fe：塗りつぶしの有無（FILL or EMPT）  */
void YCircle(int x, int y, int r, unsigned
long c, int fe) {
  int xx, yy, ww, hh;
  xx = x - r;
  yy = y - r;
  ww = 2 * r;
  hh = 2 * r;
  YForegroundColor ( c );
  XSetLineAttributes (YDisp, Ygc, 1,
  LineSolid, CapButt, JoinMiter);
  if ( fe == 1) {
     XFillArc (YDisp, Ywin, Ygc, xx, yy,
     ww, hh, 0, 360*64);
   }
  else {
     XDrawArc (YDisp, Ywin, Ygc, xx, yy,
     ww, hh, 0, 360*64);
   }
  XFlush(YDisp);
}

/*  長方形の描画                          */
/*     x：左上端の頂点のｘ座標            */
/*     y：左上端の頂点のｙ座標            */
/*     w：幅                              */
/*     h：高さ                            */
/*     c：色                              */
/*    fe：塗りつぶしの有無（FILL or EMPT）  */
void YRectangle(int x, int y, int w, int h,
unsigned long c, int fe) {
  YForegroundColor ( c );
  XSetLineAttributes (YDisp, Ygc, 1,
  LineSolid, CapButt, JoinMiter);
  if ( fe == 1) {
     XFillRectangle (YDisp, Ywin, Ygc, x, y, w, h);
    }
  else {
     XDrawRectangle(YDisp, Ywin, Ygc, x, y, w, h);
    }
  XFlush(YDisp);
}
/*  フォアグランド色の設定                */
/*     c：色                              */
void YForegroundColor(unsigned long c) {
  XSetForeground(YDisp, Ygc, c);
 }
/*  使用可能色の設定            */
void YMyColor() {
  black = BlackPixel (YDisp, 0);
  white = WhitePixel (YDisp, 0);
  red   = YColor (YDisp, "red");
  blue  = YColor (YDisp, "blue");
  green = YColor (YDisp, "green");
  orange = YColor (YDisp, "orange");
  yellow = YColor (YDisp, "yellow");
  violet = YColor (YDisp, "violet");
  darkgreen = YColor (YDisp, "DarkGreen");
  purple = YColor (YDisp, "purple");
  skyblue = YColor (YDisp, "SkyBlue");
 }
/*  使用可能色の設定関数(YMyColor)で用いる関数   */
unsigned long YColor ( Display *display,
char *color) {
  Colormap cmap;
  XColor c0, c1;
  cmap = DefaultColormap (display, 0);
  XAllocNamedColor(display, cmap, color,
  &c1, &c0);
  return(c1.pixel);
}
/* 描画ウインドウの印刷，ファイルへの格納   */
void PrintOrFile() {
  int opt;
  char file_name[50], zzz[50];
  printf("=================================¥n");
  printf("=     Printing or Filing of the
Graphics Window   =¥n");
  printf("=================================¥n");
  printf("=         Print on a paper:
  Enter 1            =¥n");
  printf("=          Save in a file:
  Enter 2            =¥n");
  printf("=         Quit this routine:
  others             =¥n");
  printf("=================================¥n");
  printf(" ¥n      => ");
  scanf("%d", &opt);
  switch ( opt ) {
    case 1:
      system("xwd | xpr | lp");
      break;
    case 2:
      printf(" ¥n  Enter File Name =>");
        scanf("%s", file_name);
        strcpy( zzz, "xwd -out ");
        strcat( zzz, file_name);
        system(zzz);
      break;
       default:
      break;
    }
}
```