

**程 序 设 计 专 题**

中期project报告



**学生姓名 ： 蒋丰泽 学号： 3180101230**

**学生姓名 ： 项羽铭 学号： 3180101225**

**学生姓名 ： 竺可立 学号： 3180103773**

**2018~2019春夏学期 2019 年 5 月**

**1、前言**

C语言是一门通用计算机编程语言，应用广泛。C语言的设计目标是提供一种能以简易的方式编译、处理低级存储器、产生少量的机器码以及不需要任何运行环境支持便能运行的编程语言。尽管C语言提供了许多低级处理的功能，但仍然保持着良好跨平台的特性，以一个标准规格写出的C语言程序可在许多电脑平台上进行编译，甚至包含一些嵌入式处理器（单片机或称MCU）以及超级电脑等作业平台。

二十世纪八十年代，为了避免各开发厂商用的C语言语法产生差异，由美国国家标准局为C语言制定了一套完整的美国国家标准语法，称为ANSI C，作为C语言最初的标准。目前2011年12月8日，国际标准化组织（ISO）和国际电工委员会（IEC）发布的C11标准是C语言的第三个官方标准，也是C语言的最新标准，该标准更好的支持了汉字函数名和汉字标识符，一定程度上实现了汉字编程。

相比于其他语言，C语言有其自己的优势。C语言既有高级语言的特点，又具有汇编语言的特点。它可以作为系统设计语言，编写工作系统应用程序；也可以作为应用程序设计语言，编写不依赖计算机硬件的应用程序。因此，它的应用范围很广泛C语言一共只有32个关键字，9种控制语句，程序书写自由，简洁紧凑，灵活方便。C的运算符包含的范围很广泛，共有34种运算符。C语言把括号、赋值、强制类型转换等都作为运算符处理。从而使C的运算类型极其丰富，表达式类型多样化，灵活使用各种运算符可以实现在其它高级语言中难以实现的运算。此外，C的数据类型丰富，有如整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等，能用来实现各种复杂的数据类型的运算，并引入了指针概念,使程序效率更高。

虽然C语言也是强类型语言，但它的语法比较灵活，允许程序编写者有较大的自由度。而且 C语言允许直接访问物理地址，可以直接对硬件进行操作。因此C既具有高级语言的功能，又具有低级语言的许多功能，能够像汇编语言一样对位、字节和地址进行操作,而这三者是计算机最基本的工作单元，可以用来写系统软件。不仅如此， C语言有一个突出的优点就是适合于多种操作系统,如DOS、UNIX,也适用于多种机型。另外C语言具有强大的图形功能,支持多种显示器和驱动器，且计算功能、逻辑判断功能强大。例如我们小组此次想要完成的贪吃蛇程序，就可以通过C语言与VS软件的合作实现对图形的处理。

# 2、项目功能目标

实现一个经典的贪吃蛇小游戏，实现开始游戏（键盘操作读取），游戏内控制贪吃蛇前进方向（键盘操作），菜单功能与图表工具栏（菜单系统与鼠标操作读取），快捷键，状态信息栏（显示当前游戏状态以及分数等信息），判断胜利条件以及失败条件等等。

# 3、总体架构和主要算法

使用libgraphics图形库绘制图形用户界面，libgraphics图形库的功能经过老师们的修改与添加，功能已经较为全面，虽然无法引用外部图片文件的内容，但是功能已经较为全面，其中的各种函数已经能够支持较为简单的图形用户界面程序编写，实现绘制基本图形与读取用户键鼠操作等功能，且其中的各个函数都较为浅显易懂，方便使用。以下是运用到的函数以及其用法。

graphics.h Overview

这个接口提供了对一个简单的函数库的访问，这样就可以在屏幕上绘制线条和弧线。这个接口提供了一个可移植的抽象化，可以与在不同硬件平台上实现的各种窗口系统一起使用。该库提供了几个函数，用于在屏幕上的一个区域绘制线条和圆弧，该区域被定义为“图形窗口”。一旦绘制，这些线和弧线停留在他们的位置，这意味着包只能用于静态图片，而不是动画。

窗口内的各个点是通过给出它们的x和y坐标来指定的。这些坐标是以英寸为单位的实数，原点在左下角，就像在传统数学中一样。

\* Function: InitGraphics

\* Usage: InitGraphics();

此过程在屏幕上创建图形窗口。对InitGraphics的调用必须在调用此包中的其他函数之前进行，

并且必须在任何printf输出之前进行。在大多数情况下，InitGraphics调用是函数main中的第一个语句。

void InitGraphics(void);

void InitConsole(void);

\* Function: MovePen

\* Usage: MovePen(x, y);

此过程将当前点移动到位置(x，y)，而不画直线。

该模型是将笔从图形窗口表面拿出，然后移动到其新位置的模型。

void MovePen(double x, double y);

\* Function: DrawLine

\* Usage: DrawLine(dx, dy);

此过程通过在x方向移动笔dx英寸和在y方向移动dy英寸从当前点绘制一条直线。

最后的位置变成新的当前点。

void DrawLine(double dx, double dy);

\* Function: DrawArc

\* Usage: DrawArc(r, start, sweep);

此过程绘制一个圆弧，它总是从当前点开始。

弧本身具有半径r，并以参数Start指定的角度开始，相对于圆心。

与传统数学一样，这一角度是从沿x轴的3点钟位置逆时针方向测量的。

例如，如果开始为0，则弧从3点开始；如果开始为90，则弧从12点开始，依此类推。

绘制的圆的分数是由参数扫描指定的，参数扫描也是以度为单位的。

如果扫描为360，则DrawARC绘制一个完整的圆；如果扫描为90，则绘制四分之一圆。

如果扫描值为正，则从当前点逆时针绘制弧。如果扫描为负值，则从当前点顺时针方向画弧。

DrawARC操作结束时的当前点是笔沿弧的最终位置。

void DrawArc(double r, double start, double sweep);

\* Functions: GetWindowWidth, GetWindowHeight

\* Usage: width = GetWindowWidth();

\* height = GetWindowHeight();

这些函数返回图形窗口的宽度和高度(以英寸为单位)。

double GetWindowWidth(void);

double GetWindowHeight(void);

\* Functions: GetCurrentX, GetCurrentY

\* Usage: x = GetCurrentX();

\* y = GetCurrentY();

这些函数返回当前的x和y位置。

double GetCurrentX(void);

double GetCurrentY(void);

genlib.h Overview

该文件包含几个定义，构成由EricRoberts开发的通用ANSI C库的核心。

这个库的目标是提供一组基本的工具和约定，以提高C程序的可读性，

特别是在教学环境中使用它们时。

genlib.h提供的基本定义如下：

1.

声明几个新的“原始的”类型(最重要的是bool和String)，

这些类型作为基本类型在其他库和应用程序中使用。

2.

一组新的内存分配函数。

3.

用于错误处理的函数。

4.

具有内部出口的循环的重复语句。

\* Type: bool

此类型有两个值，false和true，分别等于0和1。

定义这种类型的大部分优点来自可读性，

因为它允许程序员提供一个变量只接受这两个值中的一个的文档。

然而，设计一个可移植的表示非常困难，因为许多库和一些编译器定义了这些名称。

定义通常是兼容的，但仍然可能被标记为错误。

\* Type: string

类型字符串与char\*类型相同，char\*传统上用于C程序。

定义新类型的主要目的是提高程序的可读性。

在使用类型字符串的抽象级别上，将字符串拆分到其组件字符中通常并不重要。

将其声明为字符串强调了这种原子性。

\* Type: stream

与字符串一样，流类型被用来提供额外的可读性，

并被定义为等效于file\*(这特别令人困惑，因为它违反了标准的大小写约定)。

这种类型不在文本中使用，而是保存在genlib.h中，因此可以在不公开任何指针的情况下教授所有CS1。

\* Constant: UNDEFINED

除了NULL之外，指针类型的唯一其他常量是未定义的，

在某些包中用作特殊的哨兵来指示未定义的指针值。

在许多这样的情况下，NULL是一个合法的数据值，因此不适合作为哨兵。

\* Function: GetBlock

\* Usage: ptr = (type) GetBlock(nbytes);

GetBlock分配给定大小的内存块。如果没有可用内存，GetBlock将生成一个错误。

void \*GetBlock(size\_t nbytes);

\* Function: FreeBlock

\* Usage: FreeBlock(ptr);

FreeBlock释放与PTR关联的内存，该内存必须使用GetBlock、New或NewArray分配。

void FreeBlock(void \*ptr);

\* Macro: New

\* Usage: p = New(pointer-type);

New伪函数分配足够的空间来保存指针类型所指向的对象，

并将指针返回到新分配的指针。请注意，“New”与C中使用的“New”运算符不同；

前者采用指针类型，后者采用目标类型。

\* Macro: NewArray

\* Usage: p = NewArray(n, element-type);

NewArray分配足够的空间来容纳指定元素类型的n个值的数组。

\* Function: Error

\* Usage: Error(msg, ...)

Error生成一个错误字符串，展开显示在错误消息字符串中的%结构，

就像printf一样。如果引入了错误处理程序异常(请参阅“Exction.h”工具)，

则会以扩展的错误字符串作为参数引发ErrorException异常。

如果没有定义ErrorException，则程序退出时会有指示失败的状态代码

(由常量ErrorExitStatus提供)。展开后的错误消息字符串的长度不能超过MaxErrorMessage，

客户端有责任确保这一点。

void Error(string msg, ...);

extgraph.h Overview

这个接口是扩展的图形界面。它包括Graphics.h中的所有设施，加上设计用于支持更复杂、更交互式图形的其他几个功能。

\* Function: DrawEllipticalArc

\* Usage: DrawEllipticalArc(rx, ry, start, sweep);

此过程绘制椭圆弧。它的操作与图形界面中的DrawARC完全相同，

只是半径沿两个轴是不同的。

void DrawEllipticalArc(double rx, double ry, double start, double sweep);

\* Functions: StartFilledRegion, EndFilledRegion

\* Usage: StartFilledRegion(density);

\* . . . other calls . . .

\* EndFilledRegion();

这些调用使在显示器上绘制填充形状成为可能。

调用StartFilledRegion之后，对DrawLine和DrawARC的任何调用都用于创建形状定义，

并且在调用EndFilledRegion之前不会出现在屏幕上。

线条和弧必须是连续的，也就是说每个新元素必须从最后一个元素结束的位置开始。

MovePen调用可能发生在区域的开始或结束，但不在内部。

当调用EndFilledRegion时，整个区域出现在屏幕上，其内部被填充。

密度参数是介于0和1之间的一个数字，并指示如何将点密度用于填充模式。

如果密度为1，形状将以实色填充；如果为0，填充将是不可见的。

在这两者之间，实现将使用一个点模式，颜色的一些屏幕点，而不是其他。

（备注：函数StartFilledRegion在参数density为1时会画出空心的而非实心的区域，这可能是函数内部bug导致的，建议用接近1的浮点数如0.99代替1。）

void StartFilledRegion(double density);

void EndFilledRegion(void);

\* Function: DrawTextString

\* Usage: DrawTextString(text);

此函数在当前字体和大小的当前点显示字符串文本。

更新当前点，以便下一个DrawTextString命令从下一个字符位置继续。

字符串可能不包括换行符。

void DrawTextString(string text);

\* Function: TextStringWidth

\* Usage: w = TextStringWidth(text);

如果以当前字体和大小显示，此函数将返回文本字符串的宽度。

double TextStringWidth(string text);

\* Function: SetFont

\* Usage: SetFont(font);

这个函数根据字体字符串设置一个新的字体，

这是独立于案例的.不同的系统支持不同的字体，

虽然像“Times”和“Courier”这样的常用字体经常被支持。

最初，字体被设置为“Default”，这是始终支持的，尽管基础字体是系统依赖的。

如果无法识别字体名，则不会生成错误，字体保持不变。

如果需要检测此条件，可以调用GetFont查看更改是否生效。

在这种情况下，通过不生成错误，程序变得更加可移植。

void SetFont(string font);

\* Function: GetFont

\* Usage: font = GetFont();

此函数将当前字体名称作为字符串返回。

string GetFont(void);

\* Function: SetPointSize

\* Usage: SetPointSize(size);

此函数设置一个新的点大小。如果特定字体不支持点大小，则选择最近的现有大小。

void SetPointSize(int size);

\* Function: GetPointSize

\* Usage: size = GetPointSize();

此函数返回当前点大小。

int GetPointSize(void);

\* Function: SetStyle

\* Usage: SetStyle(style);

此函数基于参数样式为文本建立当前样式属性，

该参数样式是表示任何文本样式常量之和的整数。

#define Normal 0

#define Bold 1

#define Italic 2

void SetStyle(int style);

\* Function: GetStyle

\* Usage: style = GetStyle();

此函数返回当前样式。

int GetStyle(void);

\* Functions: GetMouseX, GetMouseY

\* Usage: x = GetMouseX();

\* y = GetMouseY();

这些函数分别返回鼠标的x和y坐标。

坐标值是实数，以英寸为单位，从原点测量，因此与绘图坐标相匹配。

double GetMouseX(void);

double GetMouseY(void);

\* Functions: MouseButtonIsDown

\* Usage: if (MouseButtonIsDown()) . . .

如果鼠标按钮当前处于按下状态，则此函数返回true。

为了实现间的最大兼容性，假定鼠标有一个按钮。

如果鼠标有多个按钮，则如果任何按钮按下，此函数将返回true。

bool MouseButtonIsDown(void);

\* Functions: WaitForMouseDown, WaitForMouseUp

\* Usage: WaitForMouseDown();

\* WaitForMouseUp();

WaitForMousedown函数等待鼠标按钮按下，然后返回。

WaitForMouseUp等待按钮被释放。

void WaitForMouseDown(void);

void WaitForMouseUp(void);

\* Function: HasColor

\* Usage: if (HasColor()) . . .

如果图形窗口可以显示彩色图像，则此函数返回true。

请注意，此条件比简单地检查颜色显示是否可用要强。

因为彩色窗口比黑白窗口需要更多的内存，

如果没有足够的内存来存储彩色图像，

此函数将在彩色屏幕上返回false。

例如，在Macintosh上，在创建彩色窗口之前，通常需要将分区大小增加到至少1MB。

bool HasColor(void);

\* Function: SetPenColor

\* Usage: SetPenColor(color);

此函数设置用于任何绘图的钢笔的颜色，包括线条、文本和填充区域。

颜色是一个字符串，通常是下列预定义的颜色名称之一：

Black, Dark Gray, Gray, Light Gray, White,

Red, Yellow, Green, Cyan, Blue, Magenta

黑色，深色灰色，浅灰色，白色，红色，黄色，绿色，青色，蓝色，洋红色

第一行对应于标准灰度，第二行对应于光的原色和第二种颜色。

内置集仅限于这些颜色，因为它们在所有硬件设备上可能是相同的。

对于更精细的颜色控制，您也可以使用DefineColor函数创建新的颜色名称。

void SetPenColor(string color);

\* Function: GetPenColor

\* Usage: color = GetPenColor();

此函数以字符串的形式返回当前的钢笔颜色。

string GetPenColor(void);

\* Function: SetPenSize

\* Usage: SetPenSize(size);

此函数设置用于任何绘图的笔的大小(以像素为单位)。

void SetPenSize(int size);

\* Function: GetPenSize

\* Usage: size = GetPenSize();

此函数返回用于任何绘图的笔的大小(以像素为单位)。

int GetPenSize(void);

\* Function: DefineColor

\* Usage: DefineColor(name, red, green, blue);

此函数允许客户端通过提供红色、绿色和蓝色的亮度级别来定义新的颜色名称，

这些颜色是光的主要颜色。颜色值以0到1之间的实数形式提供，表示该颜色的强度。

DefineColor允许您在许多显示器上创建中间颜色，尽管结果因硬件而异。

void DefineColor(string name, double red, double green, double blue);

\* Function: SetEraseMode

\* Usage: SetEraseMode(TRUE);

\* SetEraseMode(FALSE);

SetEraseMode函数设置内部擦除标志的值。

设置此标志的效果类似于将颜色设置为“白色”，但不影响当前的颜色设置。

当擦除模式设置为false时，将使用当前颜色还原正常绘图。

void SetEraseMode(bool mode);

\* Function: GetEraseMode

\* Usage: mode = GetEraseMode();

此函数返回擦除模式标志的当前状态。

bool GetEraseMode(void);

\* Function: SetWindowTitle

\* Usage: SetWindowTitle(title);

该函数设置图形窗口的标题，如果可以在显示器上进行这样的操作。

如果无法实现特定的实现，则只忽略调用。

此函数可以在InitGraphics调用之前调用，以设置窗口的初始名称。

void SetWindowTitle(string title);

\* Function: GetWindowTitle

\* Usage: title = GetWindowTitle();

此函数返回图形窗口的标题。如果实现不支持标题，则此调用将返回空字符串。

string GetWindowTitle(void);

\* Function: UpdateDisplay

\* Usage: UpdateDisplay();

此函数启动图形窗口的立即更新，是动画所必需的。

通常，只有当程序等待用户输入时，图形窗口才会更新。

void UpdateDisplay(void);

\* Function: Pause

\* Usage: Pause(seconds);

暂停函数更新图形窗口，然后暂停指定的秒数。

这一功能是有用的动画，否则运动会太快。

void Pause(double seconds);

\* Function: ExitGraphics

\* Usage: ExitGraphics();

ExitGraphics函数关闭图形窗口并退出应用程序，而无需等待任何其他用户交互。

void ExitGraphics(void);

\* Functions: SaveGraphicsState, RestoreGraphicsState

\* Usage: SaveGraphicsState();

\* . . . graphical operations . . .

\* RestoreGraphicsState();

SaveGraphicsState函数内部保存当前图形状态(当前的笔位置、字体、点大小和擦除模式标志)，

以便通过下一个RestoreGraphicsState调用恢复它们。

这两个函数必须成对使用，但可以嵌套到任意深度。

void SaveGraphicsState(void);

void RestoreGraphicsState(void);

\* Functions: GetFullScreenWidth, GetFullScreenHeight

\* Usage: width = GetFullScreenWidth();

\* height = GetFullScreenHeight();

这些函数返回整个显示屏幕的高度和宽度，而不是图形窗口。

它们唯一的重要用途是应用程序需要根据可用的屏幕空间调整图形窗口的大小。

在调用InitGraphics之前，可以调用这些函数。

double GetFullScreenWidth(void);

double GetFullScreenHeight(void);

\* Functions: SetWindowSize

\* Usage: SetWindowSize(width, height);

如果可能，此函数将窗口大小设置为指定的维度。

此函数应在InitGraphics创建图形窗口之前调用。

大多数实现忽略更改现有窗口大小的尝试。

应该谨慎使用此函数，因为它降低了应用程序的可移植性，

特别是当客户端请求的空间超过屏幕上可用的空间时。

void SetWindowSize(double width, double height);

\* Functions: GetXResolution, GetYResolution

\* Usage: xres = GetXResolution();

\* yres = GetYResolution();

这些函数返回每英寸沿每个坐标方向的像素数，对于应用程序非常有用，

在这些应用程序中，必须用点间距来统一表示短距离。

即使x和y分辨率对于大多数显示器是相同的，客户端也不应该依赖这个属性。

注意：图形库中的线条是单位宽度的一个像素，

其长度总是比您预期的长一个像素。例如，函数调用

DrawLine(2 / GetXResolution(), 0);

从当前点到点再往右画一条线，这样就形成了长度为一条三像素的线。

double GetXResolution(void);

double GetYResolution(void);

/\*pixels to inches\*/

double ScaleXInches(int x);

double ScaleYInches(int y);

自定义函数

void InitGameBox(void)

用于初始化游戏界面

Void SetFood(void)

用于随机设置食物位置

Void Eat(double x, double y)

用于增加节点

Void Game(double x, double y, int f)

用于游戏操作

程序使用构建链表的方式储存贪吃蛇身体节点位置，通过增加节点表示贪吃蛇吃掉食物之后增加身体长度。用整型变量f储存贪吃蛇行进方向。程序使用libgraphics图形库提供的随机函数产生不位于边界的初始点以及不处于蛇身上的食物位置，使用图形库提供的键鼠操作读取函数读取用户对游戏的操作。目前仍然缺少最高记录的变量以及成绩输出生成文件的功能，有待进一步改进。

**程序编写过程中发现的问题以及可能的解决方法：**

在程序编写过程中，发现extgraph.h头文件中定义的函数StartFilledRegion在参数density为1时会画出空心的而非实心的区域，这可能是函数内部bug导致的，建议用接近1的浮点数如0.99代替1。

在程序编写过程中，发现extgraph.h头文件中定义的函数Pause在主程序中不起作用，其错误原因有待进一步检查。使用定时器timer相关函数代替相关功能之后需求得到解决，但Pause函数存在的问题依旧没有得出结论。

在程序编写过程中，发现在使用系统头文件库中的windows.h头文件中定义的函数sleep以及函数system(“PAUSE”)时图形界面会出现未响应的情况，这可能需要函数UpdateDisplay解决问题，具体方法有待进一步探索。

# 4、人员职责

指出小组相关人员的角色和职责，包括名称、对应人员姓名以及职责。

|  |  |  |
| --- | --- | --- |
| 角色 | 姓名（学号） | 职责 |
| 组长 | 蒋丰泽  3180101230 | * 总体策划，负责协调活动 * 负责召集组内成员学习讨论 * IDE下载、安装、调试； * 库的下载、安装、使用； * 项目需求确定； * 项目总体架构设计和主要算法设计； * Project中期报告撰写。 |
| 组员 | 项羽铭  3180101225 | * IDE下载、安装、调试； * 库的下载、安装、使用； * 项目需求确定； * 项目总体架构设计和主要算法设计； * Project中期报告撰写。 |
| 组员 | 竺可立  3180103773 | * IDE下载、安装、调试； * 库的下载、安装、使用； * 项目需求确定； * 项目总体架构设计和主要算法设计； * Project中期报告撰写。 |

成员间使用github共同开发项目，每个人都有一定的贡献度。

# 5、上传要求

要求同一组的中期project报告组长负责提交（文件名为“中期report\_组长姓名.pdf”），于夏学期第一周之前上传至： pintia.cn。

# 6、附录

（体现团队合作方面的证据，如小组会议记录等体现良好的沟通能力，如邮件关键内容屏幕截图）

小组会议记录：

与公司联系邮件：