# C#语言实现

## 1.C# 函数 Func(string a, string b)用 Lambda 表达式怎么写?

(a,b) => {};

## 2.数列 1,1,2,3,5,8,13...第 n 位数是多少?用 C#递归算法实现

public int CountNumber(int num) {

if (num == 1 || num == 2) {

return 1;

} else {

return CountNumber(num -1) + CountNumber(num-2);

}

}

## 3.冒泡排序（手写代码）

public static void BubblingSort(int[]array) {

for (int i = 0; i < array.Length - 1; i++)

{

    for (int j = 0; j <array.Length-1-i; j++)

    {

        if (array[j] > array[j + 1])

        {

            int temp = array[j];

           array[j] = array[j + 1];

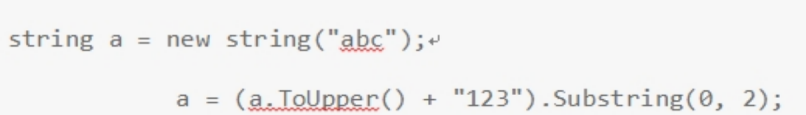
            array[j + 1] = temp;

        }

    }

}

## 4.下列代码在运行中会产生几个临时对象？\*\*



其实在 C#中第一行是会出错的（Java 中倒是可行）。应该这样初始化：

string b = new string(new char[]{'a','b','c'});

忽略错误的话：

1."abc"

2.a.ToUpper()

3."123"

4.a.ToUpper()+"123"

## 5.下列代码在运行中会发生什么问题？如何避免？



会产生运行时错误，因为 foreach 是只读的。不能一边遍历一边修改。

## 6.你拥有A块钱，一瓶水B块钱，每瓶水可以得到一个瓶盖，每C个瓶盖可以换一瓶水请写出函数求解上面题目，上面题目ABC为参数

public static int Buy(int a,int b,int c) {

return a/b + ForCap(c,a/b);

}

public static int ForCap(int c,int d) {

if (d<c) {

return 0;

} else {

return d/c + ForCap(c,d/c + d%c);

}

}

## 7.有一排开关，第一个人把所有的开关打开，第二个人按2的倍数的开关，第三个人按3的倍数的开关，以此类推，现在又n个开关，k个人，写函数求最后等两者的开关，输入参数n和k

static void Main(string[] args) {

int n = int.Parse(Console.ReadLine());

int k = int.Parse(Console.ReadLine());

Function(100,100);

}

static void Function(int n, int k) {

int i, j = 0;

bool[] a = new bool[1000]; //初始false：关灯，true:开灯

for (i = 1; i <= k; i++) //k个人

for (j = 1; j <= n; j++) //n个灯

if (j % i == 0)

a[j] = !a[j]; //取反,false变true,原来开变关，关变开

for (i = 1; i <= n; i++) //最后输出a[i]的值就可以了

if (a[i]) //灯亮着

Console.WriteLine(i);

}

## 8.数制转换，将任意整数转换成8进制形式

static void Main(string[] args) {

int n;

n =int.Parse(Console.ReadLine());

Console.WriteLine("输入的10进制为：{0}",n);

Console.Write("转换为8进制数为: ");

d2o(n);

}

static void d2o(int n) {

if (n > 7) {

d2o(n / 8);

}

Console.Write(n%8);

}

## 9.找出200以内的素数。

static void Main(string[] args) {

int count = 0;

for (int i = 1; i < 200; i++) { //外层循环：要判断的数

for (int j = 2; j <=i; j++){

if (i % j == 0&& i!=j) {

break;

}

if (j == i ) { //结束的条件:最后一个数还没有被整除

count++;

Console.WriteLine(i);

}

}

}

Console.WriteLine(count);

}

## 10.打印杨辉三角形 \*\*

public static void YHSJ(){

int [][]a= new int[7][] ;

a[0] = new int[1]; //a[0][0]=1;

a[1] = new int[2] ;

for (int i = 0; i < 7; i++) {

a[i] = new int[i+1] ;

a[i][0] =1;

a[i][i]=1;

if(i>1) { //求出中间的数据

for(int j=1;j<i; j++){

a[i][j]= a[i-1][j-1]+a[i-1][j];

}

}

}

for (int i=0; i<a.Length; i++) {

for (int k = 0; k < a.Length-1-i; k++) {

Console.Write("");

}

for(int j=0;j<a[i].Length; j++ ) {

Console.Write(a[i][j] + "");

}

Console.WriteLine();

}

}

## 11.中国有句俗话“三天打鱼两天晒网”，某人从2000年1月1日起开始“三天打鱼两天晒网”，问这个人在今后的某天中“打鱼”还是”晒网”

public static void Compute(){

Console.WriteLine ((DateTime.Now - DateTime.Parse("2000-01-01")).Days%5<3?"打鱼":"晒网");

}

## 12.假设当前市场价一只鸡10元，一只鸭12元5角。请写一个函数ShowPrice，输入参数分别为鸡和鸭的个数（非负整型），功能为显示出总价钱，精确到分。例如调用ShowPrice（5,10）后输出175.00。请注意程序的可读性和易于维护性。

static void ShowPrice(int num\_chicken, int num\_duck) {

float totalPrice = 0.00f;

float price\_chicken = 10f;

float price\_duck = 12.5f;

totalPrice = num\_chicken \* price\_chicken + num\_duck \* price\_duck;

Console.WriteLine("总价钱为：{0:0.00}", totalPrice);

}

## 13.请写一个函数，用于返回n!（阶乘）结果末尾连续0的个数，如GetZeroCount（5）返回1,因为5！ = 120，末尾连续1个0

static void Main(string[] args) {

int fac = Factorial(5);

Console.WriteLine(CountZero(fac));

}

public static int Factorial(int n) {

if (n == 1) {

return 1;

} else {

return n \* jiecheng(n - 1);

}

}

//求连续的0的个数

public static int CountZero(int num) {

int result = 0; // 最后的结果

String numStr = num.ToString();

for (int i = numStr.Length - 1; i >= 0; i--) {

if (numStr[i] == '0') {

result ++;

} else {

break;

}

}

return result;

}

## 14.有4位好友，其中一个人干了一件好事，不留名，记者来查此事，问4位，他们的回答如下：

A：不是我

B.是C

C.是D

D.他胡说

已知3个人说真话，一个人说假话，现在根据以上信息找出做了好事的人(并编程实现)

1.假设A讲假话,其馀三人讲真话,则B和C出现矛盾,所以假设不成立,A讲真话.

2.假设B讲假话,其馀三人讲真话,则C和D的

说法出现矛盾,所以假设不成立,B讲真话.

3.假设C讲假话,其馀三人讲真话,则A、B、D的证供都成立,所以假设成立,C讲假话.

4.假设D讲假话,其馀三人讲真话,则B和C出现矛盾,所以假设不成立,D讲真话.

推出,C是做好事的人.

提示：

第一步：被测者从1—4

第二步：将被测者代入表达式

第三步：判断3个表达式为真吗?是的则打印结果程序结束。不是,判断是否全测完，不是转第一步，是打印未找到，程序结束。完成下面的程序。

# include <stdio.h>

main(){

int k = 0,sum = 0,g = 0;

char man = ' ';

for(k = 1; k <= 4 && g==10;k++) {

man=64+k;

sum=(man != 'A') + (man == 'C') + (men == 'D') + (men != 'D');

if(sum==3) {

clrscr();

printf("\nman=%c\n", 64 + k);

g = 1;

}

}

if(g != 1)

printf("can't found! ");

getch();

}

# Unity实现

## 用代码实现第三角色控制器

public class Player : MonoBehaviour {

public Transform \_cameraTrans;

private Vector3 \_cameraOffset;

void Awake() {

\_cameraOffset = transform.position - \_cameraTrans.position;

}

void Update() {

\_cameraTrans.position = transform.position - \_cameraOffset;

}

}

## 写一个计时器工具，从整点开始计时，格式为：00:00:00

private float timer = 0f;

private int h = 0;

private int m = 0;

private int s = 0;

private string timeStr = string.Empty;

void Update () {

timer += Time.deltaTime;

if (timer >= 1f) {s++; timer = 0;}

if (s >= 60) {m++;s = 0;}

if (m >= 60) {h++;m = 0;}

if (h >= 99) {h = 0;}

}

void OnGUI(){

timeStr = string.Format ("{0:D2}:{1:D2}:{2:D2}", h, m, s);

GUI.Label (new Rect (10, 10, 100, 200), timeStr);

}

## 游戏中要怎么实现矩阵相乘？

//矩阵相乘

public static float[][] Mul(float[][] a, float[][] b) {

//确保矩阵a的列数和b的行数相等

if(a[0].length != b.length) {

return null;

}

//用来存放结果的矩阵，axb的结果为a的行数和b的列数

float[][] result = new float[a.length][b[0].length];

//对a的每行进行遍历

for(int i=0; i<a.length; i++) {

//对b的每列进行遍历

for(int j=0;j<b[0].length; j++) {

//c为每一个点的值

float c = 0;

//第i行j列的值为a的第i行上的n个数和b的第j列上的n个数对应相乘之和，其中n为a的列数，也是b的行数，a的列数和b的行数相等

for(int k=0; k<a[0].length; k++) {

c += (a[i][k]\*b[k][j]);

}

result[i][j] = c;

}

}

return result;

}

## 用鼠标实现在场景中拖动物体，用鼠标滚轮实现缩放(用一个 Cube 即可)。

在场景中添加一个Plan，Camera，Directional Light，Cube。添加两个脚本scrollerScirpt（挂在Camera），CubeDragScript（挂在Cube上）。

1.鼠标滚轮实现缩放：将摄像机的镜头拉近或者拉远，调整摄像机的视角就可以实现，主要实现代码如下：

void Update () {

//鼠标滚轮的效果

if (Input.GetAxis("Mouse ScrollWheel") < 0) {

if (Camera.main.fieldOfView <= 100)

Camera.main.fieldOfView += 2;

if (Camera.main.orthographicSize <= 20)

Camera.main.orthographicSize += 0.5F;

}

//Zoom in

if (Input.GetAxis("Mouse ScrollWheel") > 0) {

if (Camera.main.fieldOfView > 2)

Camera.main.fieldOfView -= 2;

if (Camera.main.orthographicSize >= 1)

Camera.main.orthographicSize -= 0.5F;

}

}

2.鼠标实现在场景中拖动物体：

　　解决思路就是将世界坐标转换成屏幕坐标，然后计算物体与鼠标之间移动量，循环鼠标被按下操作，得到鼠标的当前位置，加上计算好的移动量，将新的坐标赋值给物理就行了。主要是开启一个协同程序（Corountine）来处理

主要代码如下：

// Use this for initialization

void Start () {

StartCoroutine(OnMouseDown());

}

IEnumerator OnMouseDown() {

//将物体由世界坐标系转换为屏幕坐标系

Vector3 screenSpace = Camera.main.WorldToScreenPoint(transform.position);

//完成两个步骤 1.由于鼠标的坐标系是2维，需要转换成3维的世界坐标系

//2.只有3维坐标情况下才能来计算鼠标位置与物理的距离，offset即是距离

//将鼠标屏幕坐标转为三维坐标，再算出物体位置与鼠标之间的距离

Vector3 offset = transform.position - Camera.main.ScreenToWorldPoint(new Vector3(Input.mousePosition.x, Input.mousePosition.y, screenSpace.z));

while (Input.GetMouseButton(0)) {

//得到现在鼠标的2维坐标系位置

Vector3 curScreenSpace = new Vector3(Input.mousePosition.x, Input.mousePosition.y, screenSpace.z);

//将当前鼠标的2维位置转换成3维位置，再加上鼠标的移动量

Vector3 curPosition = Camera.main.ScreenToWorldPoint(curScreenSpace) + offset;

//curPosition就是物体应该的移动向量赋给transform的position属性

transform.position = curPosition;

yield return new WaitForFixedUpdate(); //这个很重要，循环执行

}

}

## 空间内一物体绕球面固定点（0、1、2）按照固定速度speed环绕运动。

public float Speed = 1;

void Update () {

transform.RotateAround (new Vector3(0,1,2),Vector3.up,Speed);

}

## 写一个角色控制器，鼠标控制屏幕晃动，鼠标控制开枪。

public class Player : MonoBehaviour {

public GameObject \_prefabBullet;

private float \_angleSpeed = 120f;

void Update() {

float eularY = Input.GetAxis("Mouse X") \* \_angleSpeed \* Time.deltaTime;

transform.Rotate(new Vector3(0, eularY, 0));

if (Input.GetMouseButtonDown(0)) {

Instantiate(\_prefabBullet, transform.position, transform.rotation);

}

}

}

## 3D空间有三个cube当做点，有一条鱼的模型，要求在三点之间游动，要求转向平滑一点，控制鱼的运动朝向（用四元数和欧拉角）

使用transform.localRotation = Quaternion.Slerp(Quaternion a,Quaternion b,float c)实现物体平滑转向

## 1.写一个角色控制器，鼠标控制屏幕晃动，鼠标控制开枪。

**2.敌人AI，有各种状态，实现各种状态之间的切换。**

**3.敌人会和主角对抗，敌人被打到之后，会闪一次红色，然后红色比例提升10%，10次攻击之后，成红色。**

**4.敌人会自动攻击主角，主角也会有颜色变化。**

**5.敌人会在范围内巡逻。**

**6.UI，左边显示8个AI的被攻击次数，右边显示AI的攻击次数排序。**

public class Player : MonoBehaviour {

public Camera \_camera;

public GameObject \_prefabBullet;

private float \_angleSpeed = 120f;

//生命值

private int \_life = 10;

//玩家的状态

private bool \_state = false;

//是否被打到

public void IsState() {

if (\_state && \_life > 0) {

\_life -= 1;

\_state = false;

} else {

\_state = true;

}

}

public void RoleRotate() {

float eularY = Input.GetAxis("Mouse X") \* \_angleSpeed \* Time.deltaTime;

}

//风发射子弹

public void RoleShoot() {

if (Input.GetMouseButtonDown(0)) {

Instantiate(\_prefabBullet, transform.position, transform.rotation);

}

}

}

public class Enemy : MonoBehaviour {

//敌人被打到状态

private bool \_state = false;

//玩家与敌人距离

private float \_distance;

//角色

public GameObject \_role;

//生命值

private int \_life = 10;

//敌人被攻击次数

private int \_timeAccack = 0;

//敌人状态

public void EnemyState() {

if (\_distance >= 10f) {

if (\_life >= 1 && \_state == true) {

\_life -= 1;

\_timeAccack++;

}

\_state = false;

}

else if (\_distance >= 0 && \_state == false) {

\_state = true;

}

}

//敌人与玩家距离

public void Distance() {

\_distance = Vector3.Distance(transform.position, \_role.transform.position);

}

void OnGUI() {

GUI.TextArea(new Rect(20, 50, 80, 30), \_timeAccack.ToString());

GUI.TextArea(new Rect(20, 90, 80, 30), \_timeAccack.ToString());

GUI.TextArea(new Rect(20, 130, 80, 30), \_timeAccack.ToString());

}

}

# 设计模式实现

## 1.单例模式

public class Singleton {

private static Singleton instance;

private Singleton (){}

public static synchronized Singleton getInstance() {

if (instance == null) {

instance = new Singleton();

}

return instance;

}

}

## 2.观察者模式

class Program

{

// 委托充当订阅者接口类

public delegate void NotifyEventHandler(object sender);

// 抽象订阅号类

public class Blog

{

public NotifyEventHandler NotifyEvent;

public string Symbol { get; set; }//描写订阅号的相关信息

public string Info { get; set; }//描写此次update的信息

public Blog(string symbol, string info)

{

this.Symbol = symbol;

this.Info = info;

}

#region 新增对订阅号列表的维护操作

public void AddObserver(NotifyEventHandler ob)

{

NotifyEvent += ob;

}

public void RemoveObserver(NotifyEventHandler ob)

{

NotifyEvent -= ob;

}

#endregion

public void Update()

{

if (NotifyEvent != null)

{

NotifyEvent(this);

}

}

}

// 具体订阅号类

public class MyBlog : Blog

{

public MyBlog(string symbol, string info)

: base(symbol, info)

{

}

}

// 具体订阅者类

public class Subscriber

{

public string Name { get; set; }

public Subscriber(string name)

{

this.Name = name;

}

public void Receive(Object obj)

{

Blog xmfdsh = obj as Blog;

if (xmfdsh != null)

{

Console.WriteLine("订阅者 {0} 观察到了{1}{2}", Name, xmfdsh.Symbol, xmfdsh.Info);

}

}

}

static void Main1(string[] args)

{

Blog xmfdsh = new MyBlog("xmfdsh", "发布了一篇新博客");

Subscriber wnm = new Subscriber("王尼玛");

Subscriber tmr = new Subscriber("唐马儒");

Subscriber wmt = new Subscriber("王蜜桃");

Subscriber anm = new Subscriber("敖尼玛");

// 添加订阅者

xmfdsh.AddObserver(new NotifyEventHandler(wnm.Receive));

xmfdsh.AddObserver(new NotifyEventHandler(tmr.Receive));

xmfdsh.AddObserver(new NotifyEventHandler(wmt.Receive));

xmfdsh.AddObserver(new NotifyEventHandler(anm.Receive));

xmfdsh.Update();

Console.WriteLine();

Console.WriteLine();

Console.WriteLine();

Console.WriteLine();

Console.WriteLine("移除订阅者王尼玛");

xmfdsh.RemoveObserver(new NotifyEventHandler(wnm.Receive));

xmfdsh.Update();

Console.ReadLine();

}

}

<https://www.cnblogs.com/xmfdsh/p/4047114.html>

## 3.工厂模式

工厂的接口

public abstract class IFactory

{

//生产 抽象方法

public abstract void Production();

}

工厂可实现的产品,鞋子,衣服

//生产鞋子

public class ShoesClass : IFactory

{

public override void Production()

{

Console.WriteLine("生产鞋子");

}

}

public class ClothesClass : IFactory

{

public override void Production()

{

Console.WriteLine("生产了衣服");

}

}

根据不同需求来实现工厂的生产功能

public class Factory

{

IFactory f = null;

public IFactory Create(string type)

{

switch (type)

{

case "Shoes":

f = new ShoesClass();

break;

case "Clothes":

f = new ClothesClass();  
  
break;

default:

break;

}

return f;

}

}

客户端实现

class Program

{

static void Main(string[] args)

{

IFactory f = new Factory().Create("Clothes");

f.Production();

Console.ReadKey();

}

}

<https://www.cnblogs.com/heroysj/p/10875046.html>

# 四．Shader实现

## 1.如何实现以下人物在树丛中部分透明效果？

Shader "Custom/PlayerDiffuse" {

Properties {

\_NotVisibleColor ("NotVisibleColor (RGB) ", Color) = (0.3,0.3,0.3,1)

\_MainTex ("Base (RGB) ", 2D) = "white"{}

}

SubShader {

Tags { "Queue"= "Geometry+500""RenderType"="Opaque"}

LOD 200

Pass {

ZTest Greater

Lighting Off

ZWrite Off

//Color [\_NotVisibleColor]

Blend SrcAlpha OneMinusSrcAlpha

SetTexture [\_MainTex] { ConstantColor [\_NotVisibleColor] combine constant \* texture }

}

Pass {

ZTest LEqual

Material {

Diffuse (1,1,1,1)

Ambient (1,1,1,1)

}

Lighting Off

SetTexture [\_MainTex] { combine texture }

}

}

FallBack "Diffuse"

}

## 2.Shader的代码实现？大概写一下

Shader "Custom/NewSurfaceShader"{

Properties {

\_Color ("Color", Color) = (1,1,1,1)

\_MainTex ("Albedo (RGB) ", 2D) = "white" {}

\_Glossiness ("Smoothness", Range(0,1)) = 0.5

\_Metallic ("Metallic", Range(0,1)) = 0.0

}

SubShader {

Tags { "RenderType"="Opaque" }

LOD 200

CGPROGRAM

// Physically based Standard lighting model, and enable shadows on all light types

#pragma surface surf Standard fullforwardshadows

// Use shader model 3.0 target, to get nicer looking lighting

#pragma target 3.0

sampler2D \_MainTex;

struct Input {

float2 uv\_MainTex;

};

half \_Glossiness;

half \_Metallic;

fixed4 \_Color;

void surf (Input IN, inout SurfaceOutputStandard o) {

// Albedo comes from a texture tinted by color

fixed4 c = tex2D (\_MainTex, IN.uv\_MainTex) \* \_Color;

o.Albedo = c.rgb;

// Metallic and smoothness come from slider variables

o.Metallic = \_Metallic;

o.Smoothness = \_Glossiness;

o.Alpha = c.a;

}

ENDCG

}

FallBack "Diffuse"

}

# Lu热更新实现

lua 的热更新机制主要由require（）,来决定通过package.loaded[moudle\_name] = nil 强制删除，当前内存中的，然后require（）加载该模块，但是这里有个问题，这样的热更新只能热更新新的引用，所以先用一个值保存旧的模块，再引用的新的模块逐渐替换，之前引用该模块的文件可以热更新掉

进行热更新就是先查找版本号，如果版本号不一致，就会查找那些资源包的MD5码发生了改变，根据那些不同的MD5码的资源包，把新的资源包名称换成元表的资源包名称，然后遍历新的资源包信息对照旧的资源信息进行替换。

function reload\_moudle(moudle\_name)

local old\_name = \_G[moudle\_name] --保存引用的

package.loaded[moudle\_name] = nil

require(moudle\_name) --加載新的

local new\_name = \_G[moudle\_name]

for k,v in pairs(new\_name) do

old\_name[k] =v --引用成新的

end

package.loaded[moudle\_name] = old\_name --重新添加

End

# 对象池实现

# 协程实现

# 状态机实现

## 1.设计一个状态机类型，状态值为int类型，要求：

拥有接口，获取当前状态，切换状态

外部可以监听状态切换事件，参数为切换前状态和切换后状态（使用delete和event）//定义一个状态

public abstract class IAction{

public int StateName;

public IAction(int stateName) {

this.StateName = stateName;

}

public int GetState(){

return StateName;

}

public abstract bool CanGetIn();

public abstract void GetIn();

public abstract void GetOut();

public abstract void Update(float dt);

}

//定义一个状态切换事件

public abstract class IEvent {

public int code;

public IEvent(int code) {

this.code = code;

}

public int GetCode(){

return code;

}

//当前状态事件检测

public abstract bool Check();

}

public class AvatarStateMachine{

//初始化状态机

public void InitStateMachine(){}

//注册一个状态

public void AddAction(int action) {}

//注册一个状态切换事件

public void AddEventTransition(int fromAction, int toAction, IEvent byEvent) {}

//更新当前状态，检测能否进入下一个状态

public void UpdateStateMachine(){}

//强行切换状态

public void SwitchTo(int toState) {}

}

# A星算法实现

## 1.A星算法解释

首先是先把地图分成区域块，然后利用OPEN,CLOSED列表进行选择添加区域块。每通过一个区域块给每一个区域块一个和值，该值被称为路径增量。在角色进行下一步运动时会判断添加进来的open列表的区域块的F值，之后会根据最低的和值，网目标点移动。

一个记录下所有被考虑来寻找最短路径的方块（称为open 列表）

一个记录下不会再被考虑的方块（成为closed列表）

列出了公式F = G + H 中的每项值

F（方块的和值）：左上角

G（从A点到方块的移动量）：左下角

H（从方块到B点的估算移动量): 右下角

<https://www.iteye.com/blog/longpo-2232330>

# 心跳包实现

## Socket之心跳包实现思路

客户端连接上服务端后，在服务端会维护一个在线客户端列表。客户端每隔一段时间，向服务端发送一个心跳包，服务端受收到包以后，会更新客户端最近一次在线时间。一旦服务端超过规定时间没有接收到客户端发来的包，则视为掉线。

一分钟一次

https://www.cnblogs.com/snaildev/p/7724867.html

# 杂篇

## 十四．你会打包吗

1anriond

通过build setting选择不同平台，然后把产品名称和公司名称改成一致，在playsetting调整相应的需求，比如品牌动画、产品图标等。然后设置好Build就可以打包成功了。

1. IOS

先决条件是必须为苹果mac机。拥有公司苹果账号，并确保电脑上安装了unity，unity包 ios-support。和xcode。

打开了unity应用之后，选择buildSettings

配置应用相关图标显示，以及其他需要的选项，注意此处填写的bundleID

点击打开刚打包好的文件夹，选择.xcodeproj文件，双击打开

这里注意，如果是要测试，请在官网上把测试苹果机的机器信息填写进去，并在这里选择iphone测试，如果是直接要打包的话，请选择Generic IOS Device

出现项目相关的配置信息。确认版本中与unity中的一致，并在Signing中的Team中填写公司的苹果账号

打包完成后会自动弹出文件夹，点击打包好的应用，点击Export

然后从网上下载iReSign重签名工具，

1）第一行填写上个图中ipa包的路径

2）第二行填写从苹果官网自己生成的签名文件

3）第三行不要进行填写

4）输入在苹果官网注册证书所使用的苹果id

5）第五行选择相对应的苹果证书

IOS打包流程<https://www.cnblogs.com/mzwl/p/6724034.html>

3.AB包

我们先在assets文件夹下新建一个文件夹（Editor）,文件夹的名字一定要叫：Editor

然后新建一个脚本定好存储位置。

然后BulidAssetBundles就打包好了

加载可以通过内存同步异步和本地异步以及服务器来搜索文件进行加载。

## 十六.你会使用SDK吗

将 Unity 安装目录下的 classes.jar 复制到 Android 项目的 libs 文件夹下

Android 编写代码实现

Android 导出 jar/aar

unity 导入 jar/aar

利用 AndroidJavaClass 和 AndroidJavaObject 调用相应的方法，实现功能

https://blog.csdn.net/SGamble/article/details/101376652?utm\_medium=distribute.wap\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth\_1-utm\_source=distribute.wap\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase#Unity\_\_SDK\_\_80