角色属性与相关交互函数库：pkm_character 库

一、库的组成

pkm_character.c 文件，包含所有函数的实现

pkm_character.h 文件，包含所有函数与 struct 类型的定义，类似接口。此处放置的类型与函数抽象定义皆为可被外部文件调用的函数，不在此.h 文件内定义但是又在.c 文件中实现的函数与变量请不要调用。

引入库的时候只能引入.h 文件，若引入.c 文件会报多重定义的 bug。

示例：

```
#include “pkm_character.h”
```

二、属性与可调用函数

2.1 struct 类型与属性

```
/* USER CODE BEGIN Private defines */
typedef struct character{
    int id; //识别宝可梦身份，共三种，id为0-2；id对应名字string+形象图片数组array+特殊技能等，因此此struct得以避免存任何数组
    int ATK; // 基础攻击值
    int DEF; // 基础防御值
    int HP; // 血量
    int total_HP; // 本角色总血量
    int level; //bonus 级别，打赢一次升一级，各属性+，可考虑图片变动

    //以下变量是为了方便管理特殊状态设置的，不建议使用，除非显示的图片也想跟着特殊状态一起变
    //特殊状态，为二值属性。不同角色特殊状态下有不同效果，适配需要跨回合发挥作用的技能。
    int is_active_defend; //确认角色有没有正在处于主动防御期间
    int special_state; //防御类角色是否正处于大招期间
    int weapon_state; //是否正在持有武器
    int shield_state; //是否正在持有盾牌
    int easter_state; //是否正在持有免死金牌
}cha;
```

```
        int id; //识别宝可梦身份，共三种，id 为 0-2；id 对应名字 string+形象图片数组 array+特殊技能等，因此此 struct 得以避免存任何数组
        int ATK; // 基础攻击值
        int DEF; // 基础防御值
        int HP; // 血量
        int total_HP; // 本角色总血量
        int level; //bonus 级别，打赢一次升一级，各属性+，可考虑图片变动


        //以下变量是为了方便管理特殊状态设置的，不建议使用，除非显示的图片也想跟着特殊状态一起变
        //特殊状态，为二值属性。不同角色特殊状态下有不同效果，适配需要跨回合发挥作用的技能。
        int is_active_defend; //确认角色有没有正在处于主动防御期间
        int special_state; //防御类角色是否正处于大招期间
        int weapon_state; //是否正在持有武器
        int shield_state; //是否正在持有盾牌
        int easter_state; //是否正在持有免死金牌
```

以上为结构体的所有元素变量，但其实如果没有特殊需求的话应该不会需要外部用户亲自访问此结构体的内部元素。

2.2 函数

允许被外界调用的函数定义都在.h 文件内

```c
/* USER CODE END Private defines */
struct character initialize_character(int id); // 初始化角色，返回的是struct对象
//注意！为了规避自动拷贝，以下所有函数都是传入指针而非struct本身，即如果角色变量名为“a”，则以下函数都传入“&a”，而非“a”

int get_id(struct character *p);
int get_ATK(struct character *p);
int get_DEF(struct character *p);
int get_HP(struct character *p);
int get_total_HP(struct character *p);
int get_level(struct character *p);
int is_active_defending(struct character *p);
int is_during_unique_skill_for_id_1(struct character *p);//此参数只对防御类角色有意义
int has_weapon(struct character *p);
int has_shield(struct character *p);
int has_easter(struct character *p);
void get_name(struct character *p,char name[64]);//得到角色名字，需要外部创建一个空的char数组传入本函数，稍后该数组会自动载有名字
void show_character_image(struct character *p,int x, int y);//显示角色形象，目前图片数组等待烧录
void print_basic_info(struct character *p);//在串口上打印以上角色属性与状态信息，包括存活与否

void set_ATK(struct character *p,int atk);//不建议直接修改参数，可能导致逻辑失控
void set_DEF(struct character *p,int def);//不建议直接修改参数
void set_HP(struct character *p,int hp);//不建议直接修改参数
void set_total_HP(struct character *p,int thp);//不建议直接修改参数


int normal_attack_skill(struct character *p);//普攻，立刻生效。由zzb调用
void grow_attack_skill(struct character *p);//辅助攻击技能，立刻生效。由zzb调用
void unique_skill(struct character *p);//大招，立刻生效。由zzb调用


void defend_skill(struct character *p);//由zzb调用，主动使用防御技能，使用后下次攻击触发主动防御效果，否则触发被动防御效果
void defend_clpt(struct character *p, int atk);//由hrh调用，用于放在无线通信的中断回调里，用于在被攻击时响应对方。


int is_alive(struct character *p);//由zzb调用。检查是否活着，并处理有免死金牌等特殊情况，因此请务必通过此函数检查存活而不要直接访问结构体
void prepare(struct character *p);//活着进入下一局战斗前调用，用于置零各种特效状态以及把血回满。似乎不需要这个功能但是先写了再说
void renew(struct character *p);//刷新角色，各参数全部回到初始状态。不清楚什么情况会需要这个功能，但是先写了再说


void upgrade(struct character *p);//升一级，同时置零所有特殊状态，需要zzb决定升级逻辑，由zzb调用
```

```c
void upgrade(struct character *p);//升一级，同时置零所有特殊状态，需要zzb决定升级逻辑，由zzb调用


void weapon(struct character *p);// 捡到武器，由zzb调用
void shield(struct character *p);// 捡到盾牌，由zzb调用
void aid_bag(struct character *p);// 捡到急救包，由zzb调用
void easter_egg(struct character *p);// 捡到免死金牌，由zzb调用
```

基本上分为八类函数。

2.2.1、初始化函数

根据你输入的数字初始化相应角色。一共三种角色可选，这是为了吃下 **5%“多类型角色可选”的 bonus** 做出来的。但是选角 bonus 需要 zzb 多做出一个选角界面。可能可以直接复制修改一个有特殊退出条件的电阻屏循环放在主循环之前？

```
struct character initialize_character(int id);
```

id = 0 时，返回的是偏攻击类角色的结构体，id = 1 时，返回的是偏防御类角色的结构体，id = 2 时，返回的是偏生命类角色的结构体。

需要注意，**返回的是结构体本身**，但是后续交互函数中**必须要输入结构体的整体指针**，否则 C 语言会自动拷贝结构体导致我们无法修改记录原结构体数据从而引发 bug。具体看示例。

示例：

```
a = initialize_character(0);//返回攻击类角色，变量名为 a
normal_attack_skill(&a);//a 角色发起普攻，此处输入的是 a 变量的指针，即“&a”，而非 a 本身.
```

2.2.2、get 类函数

此类函数概念对应于 java 对象的 get。此类函数封装了所有结构体的元素变量，同时配置了串口打印信息来帮助 debug，所以可以不亲自调用 struct 结构体访问元素，而是通过此类函数访问，另外，由于结构体没存图片和名字，因此想要获得名字与图片形象只能通过访问此类函数来得到。

```
int get_id(struct character *p);
```

用于得到对象的 id，但是必须要输入指针而非对象，以下代码用法类似。看示例：

```
a = initialize_character(0);//返回攻击类角色
int temp_id = get_id(&a);//输入的是“&a”，返回 id 值
```

```
int get_ATK(struct character *p);
```

输入对象指针，得到对象的攻击值

```
int get_DEF(struct character *p);
```

输入对象指针，得到对象的防御值

```
int get_HP(struct character *p);
```

输入对象指针，得到对象的当前生命值

```
int get_total_HP(struct character *p);
```

输入对象指针，得到对象的最大生命值

```
int get_level(struct character *p);
```

输入对象指针，得到对象的当前等级（此为 bonus 项目）

```
int is_active_defending(struct character *p);
```

输入对象指针，得到对象是否处于主动防御期间，是的话返回 1，否则 0。此变量用于管理防御技能状态。

```
int is_during_unique_skill_for_id_1(struct character *p);//此参数只对防御类角色有意义
```

输入对象指针，得到防御类对象是否处于防御类角色大招期间。此变量只对防御类角色有意义，因为防御类角色大招会跨回合，所以需要专门一个变量记录。另外两种角色的此变量永远不会被使用。

```
int has_weapon(struct character *p);
```

输入对象指针，得到对象是否正持有武器，是的话给 1，否则给 0。此为 bonus 项目，用于管理武器包状态，下同。具体效果在后面的函数会讲解。

```
int has_shield(struct character *p);
```

输入对象指针，得到对象是否正持有盾牌。

```
int has_easter(struct character *p);
```

输入对象指针，得到对象是否正持有免死金牌。

```
void get_name(struct character *p,char name[64]);//得到角色名字，需要外部创建一个空的 char 数组传入本函数，稍后该数组会自动载有名字
```

输入对象指针与一个空 char[64]数组，空数组将载有对象名字。由于传出一个字符数组会涉及指针操作，所以为了规避指针操作，需要你从外部创建一个空 char[64]数组传入此函数，随后此数组会自动载有对应名字。示例：

```
    char ttt[64] = {0};
    get_name(p,ttt);//随后 ttt 将自动载有相应名字。
```

```
void show_character_image(struct character *p,int x, int y);//显示角色形象，目前图片数组等待烧录
```

输入对象指针，图片的左上角坐标（x,y），随后将自动在指定位置画相对应角色的形象图。

具体逻辑参考 demo 的画图函数。但是目前还没塞入角色形象图，也还没引入需要的 lcd.h 等文件，所以此函数目前是空函数。

```
void print_basic_info(struct character *p);//在串口上打印以上角色属性与状态信息，包括存活与否
```

输入对象指针,在串口自动打印显示该对象所有的基本信息,即以上所提到的信息。这个主要用来帮忙 debug。

2.2.3、set 类函数

只开放了四个基本数值的 set 权限，其他变量因为有比较强的逻辑关联性所以不开放权限，同时也建议避免亲自 set 结构体的变量，因为后续函数都有实现对相关数值的修改功能。

```
void set_ATK(struct character *p,int atk);//不建议直接修改参数，可能导致逻辑失控
```

输入对象指针与攻击值，修改对应对象的攻击力。

```
void set_DEF(struct character *p,int def);//不建议直接修改参数
```

输入对象指针与防御值，修改对应对象的防御力。

```
void set_HP(struct character *p,int hp);//不建议直接修改参数
```

输入对象指针与血量，修改对应对象的当前血量，当前血量被限制不得超过最大血量。

```
void set_total_HP(struct character *p,int thp);//不建议直接修改参数
```

输入对象指针与血量，修改对应对象的最大血量。

2.2.4、攻击技能类函数

为了吃下 **5%“多技能”bonus**，共有三种技能。且都用串口打印了相关信息

```
int normal_attack_skill(struct character *p);//普攻，立刻生效。由 zzb 调用
```

普通攻击，传入对象指针即可发起普攻，攻击力为对象攻击值，有 40%概率会暴击，暴击率 10%。如果暴击则返回 1，没暴击则返回 0。这个输出可以不用。

但是由于还没和无线通信模块对接，所以此函数暂时不会发送任何信息，但是会往串口发送相关信息来假装发起了攻击。目前可以直接调用，后续对接代码时再补全，以下两个同。

示例：

```
a = initialize_character(1);
int is_critical = normal_attack_skill(&a);
```

```
void grow_attack_skill(struct character *p);//辅助攻击技能，立刻生效。由 zzb 调用
```

辅助攻击，输入对象指针立刻发起辅助攻击，此攻击的攻击力只有对象攻击值一半，但对象的短板属性会永久+4.

```
void unique_skill(struct character *p);//大招，立刻生效。由 zzb 调用
```

大招攻击，输入对象指针立刻发起大招攻击。此技能因角色而异：

攻击类角色：此次攻击加成 5%-50%，血越少攻击加成越多。

防御类角色：此次攻击献祭自身最大血量 5%的血，造成相当于自身防御值一半的攻击力，同时免疫下次对手的攻击；如果角色血量不够支付，则不减血，只造成相当于自身全部防御值的攻击力，下次对手攻击不能免疫。

生命类角色：此次攻击的攻击力相当于自己最大血量的 20%，同时回血，回血量相当于自身最大血量的 5%。

2.2.5、防御技能类函数

防御技能只有一个，但是加入了被动防御的被动技能。

```
void defend_skill(struct character *p);//由 zzb 调用，主动使用防御技能，使用后下次攻击触发主动防御效果，否则触发被动防御效果
```

防御技能，输入对象指针发起主动防御。由 zzb 在显示屏中调用。

下次对手攻击时，会自动触发主动防御。主动防御的防御值相当于自身全部防御值。如果不发起本技能，则对手攻击时，仍会自动触发被动防御，被动防御的防御力只有自身防御值的一半。

需要注意，防御后会自动扣除相应血量，但是并没有在内部设置检查是否死亡的函数，需要在**外部调用后续的 is_alive 函数检查角色是否死亡。请不要直接访问结构体的 HP 属性检查存活**，因为 is_alive 还包括了对持有免死金牌这种复活道具的处理，如果直接访问 HP 属性确认存活则会导致免死金牌功能失效。

```
void defend_clpt(struct character *p, int atk);//由 hrh 调用，用于放在无线通信的中断回调里，用于在被攻击时响应对方。
```

防御技能的回调函数，输入是对象指针与对手打过来的攻击值。这个是用来响应对手的攻击的，**由 hrh 他们在无线通信的中断函数里调用**。

此函数会根据角色本身是否正处于主动防御状态而选择采用相应的防御行为。**此函数过后角色可能死亡，但是本函数内不会检查角色是否死亡，请在外部调用“is_alive”检查存活情况**。

2.2.6、状态确认与重置类函数

此类函数只有 is_alive 是最重要的，另外两个可能用不上，可以忽略。

```
int is_alive(struct character *p);//由 zzb 调用。检查是否活着，并处理有免死金牌等特殊情况，因此请务必通过此函数检查存活而不要直接访问结构体
```

输入对象指针，确认对象是否存活的函数，活着返回 1，死了返回 0。

此函数等效于直接访问结构体的 HP 是否小于等于 0。但是此函数加入了对免死金牌这种复活道具的特殊处理，如果复活了也会返回 1。

所以，**检查存活请用本函数而不要直接访问结构体的 HP 变量**。

```
void prepare(struct character *p);//活着进入下一局战斗前调用，用于置零各种特效状态以及把血回满。似乎不需要这个功能但是先写了再说
```

输入对象指针，各种特殊状态会被置零，血会回满，已经提升攻击力、防御力、最大生命与等级都会保留。这个函数的本意是用来在进入下一局战斗前用的，但是我看题意似乎整个 project 就是一局胜负就结束游戏？？所以此函数可能用不上。

```
void renew(struct character *p);//刷新角色，各参数全部回到初始状态。不清楚什么情况会需要这个功能，但是先写了再说
```

输入对象指针，对象被格式化回出厂配置。可能用不上。

2.2.7、升级类函数

此类函数只有一个函数，是想吃掉 **5%“可升级”的 bonus** 弄出来的。

```
void upgrade(struct character *p);//升一级，同时置零所有特殊状态，需要 zzb 决定升级逻辑，由 zzb 调用
```

输入对象的指针，对象的等级+1，攻击力与防御力都+10，最大生命+20。同时所有特殊状态全部重置。

升级的 bonus 需要升级的逻辑，此函数只管升级，具体什么时候升级（调用本函数的时机或条件），则由 zzb 决定。

2.2.8、道具类函数

此类函数有四个函数，分别代表四种道具，是想吃掉 10%“多道具与资源”bonus 弄出来的。所有的道具都不能叠加效果，比如捡到多个武器都只能当一个武器用，除非用完后又捡到一个武器。

道具 bonus 还需要道具出现的逻辑，本函数只管提供功能。道具何时出现，如何出现，等逻辑由 zzb 决定。

```
void weapon(struct character *p);// 捡到武器，由 zzb 调用
```

输入对象指针，对象获得武器，下一次发起任何一种攻击技能时，攻击力额外+20。

```
void shield(struct character *p);// 捡到盾牌，由 zzb 调用
```

输入对象指针，对象获得盾牌，下一次被对手攻击时，防御力额外+20。

```
void aid_bag(struct character *p);// 捡到急救包，由 zzb 调用
```

输入对象指针，对象获得急救包，当前血量立刻+20，但是不会超过最大血量。

```
void easter_egg(struct character *p);// 捡到免死金牌，由 zzb 调用
```

输入对象指针，对象获得复活节彩蛋/免死金牌，可以死后复活继续战斗。复活后所有特殊状态全部重置，攻击力、防御力、最大血量与等级保留，但是复活后的当前血量只有最大血量 20%。

三、配置要求

1. **huart1 / 串口 1**:

project 并不需要使用串口功能，但是为了方便 debug，上述所有的函数都会往串口打印相关信息，所以需要配置打开串口 1（huart1）

2. lcd.h:

由于要画角色形象图，需要引入 lcd.h。但是现在那个画图函数还只是空函数，这个可以等 GUI 功能做的差不多或需要放形象图的时候再完善。

四、备注

1. 目前已经用串口通信代替无线通信 de 过 Bug，以及通过串口模拟对战来调整了攻击力防御值生命值等参数的合理性，应该不会有太大的 bug，各项数值的调整目前可以支持对战多个回合，不会出现被碾压的局面。

2. 如果把以上实现的功能（初始选角界面；三个攻击技能一个防御技能；升级；道具）全部放到显示屏中，应该能至少吃下 25%的 bonus，应该工作量不会太大，选角界面、升级条件和道具出现的逻辑可以简单一些，不需要太复杂。

复制黏贴表：

```
struct character initialize_character(int id); // 初始化角色，返回的是struct对象
//注意！为了规避自动拷贝，以下所有函数都是传入指针而非struct本身，即如果角色变量名为“a”，则以下函数都传入“&a”，而非“a”

int get_id(struct character *p);
int get_ATK(struct character *p);
int get_DEF(struct character *p);
int get_HP(struct character *p);
int get_total_HP(struct character *p);
int get_level(struct character *p);
int is_active_defending(struct character *p);
int is_during_unique_skill_for_id_1(struct character *p);//此参数只对防御类角色有意义
int has_weapon(struct character *p);
int has_shield(struct character *p);
int has_easter(struct character *p);
void get_name(struct character *p,char name[64]);//得到角色名字，需要外部创建一个空的char数组传入本函数，稍后该数组会自动载有名字
void show_character_image(struct character *p,int x, int y);//显示角色形象，目前图片数组等待烧录
void print_basic_info(struct character *p);//在串口上打印以上角色属性与状态信息，包括存活与否


void set_ATK(struct character *p,int atk);//不建议直接修改参数，可能导致逻辑失控
void set_DEF(struct character *p,int def);//不建议直接修改参数
void set_HP(struct character *p,int hp);//不建议直接修改参数
void set_total_HP(struct character *p,int thp);//不建议直接修改参数


int normal_attack_skill(struct character *p);//普攻，立刻生效。由zzb调用
void grow_attack_skill(struct character *p);//辅助攻击技能，立刻生效。由zzb调用
void unique_skill(struct character *p);//大招，立刻生效。由zzb调用


void defend_skill(struct character *p);//由zzb调用，主动使用防御技能，使用后下次攻击触发主动防御效果，否则触发被动防御效果
void defend_clpt(struct character *p, int atk);//由hrh调用，用于放在无线通信的中断回调里，用于在被攻击时响应对方。
```

```
int is_alive(struct character *p);//由 zzb 调用。检查是否活着，并处理有免死金牌等特殊情况，因此请务必通过此函数检查存活而不要直接访问结构体
void prepare(struct character *p);//活着进入下一局战斗前调用，用于置零各种特效状态以及把血回满。似乎不需要这个功能但是先写了再说
void renew(struct character *p);//刷新角色，各参数全部回到初始状态。不清楚什么情况会需要这个功能，但是先写了再说


void upgrade(struct character *p);//升一级，同时置零所有特殊状态，需要 zzb 决定升级逻辑，由 zzb 调用


void weapon(struct character *p);// 捡到武器，由 zzb 调用
void shield(struct character *p);// 捡到盾牌，由 zzb 调用
void aid_bag(struct character *p);// 捡到急救包，由 zzb 调用
void easter_egg(struct character *p);// 捡到免死金牌，由 zzb 调用
```