

- Use markdown with vim in VScode
  - How to create and update the contents?
  - How to make fenced code blocks?
  - How to make to do list?
  - How to write emoji in markdown?
  - How to add links in markdown?
- C++ learning
  - 基础篇
    - 数组
    - 函数
    - 指针
    - 结构体
      - 结构体数组
      - 结构体指针
      - 结构体嵌套
      - 结构体做函数参数
- C++核心编程
  - 内存分区模型
  - 引用
  - 函数提高
    - 函数重载
  - 类和对象
    - 封装
    - 对象的初始化和清理
    - 构造函数的分类和调用
    - 拷贝构造函数调用时机
    - 构造函数调用规则

## Use markdown with vim in VScode

---

Vim + Markdown: you can take your notes efficiently and beautifully.

Key	Command
Ctrl+b	Toggle bold
Ctrl+i	Toggle italic
Ctrl+shift+]	Toggle heading (uplevel)
Ctrl+shift+[	Toggle heading (downlevel)
Ctrl+M	Toggle math environment
Alt+C	check task list item
Ctrl+Shift+K V	Toggle preview to side
Alt+shift+f	formatter

- type "-" you can list items
  - enter to type another one
1. list items with number "1."
  2. enter for another

## How to create and update the contents?

- Use command "create table contents" to create contents
- Use "update table contents" to update the contents

## How to make fenced code blocks?

fenced code block:1 use tab

2 use three backticks if add a language next to the backticks the syntax will highlight

```
{
  from bs import BeautifulSoup
  if i++
  print "hello world"
}
```

## How to make to do list?

- ☐ to do list
- ☐ to create a task list, add dashes(-) and brackets with a space ([x]) in front of task list items.
- ☐ Alt + c and mark and unmark the task

## How to write emoji in markdown?

😊 use :: to surround a word. Such as 😂, 🍌, 😊

- 😂 joy
- 😊 smile
- 😊 blush
- ★ star
- ★ star2
- 😡 angry
- 💘 cupid
- 😍 heart\_eyes

- <https://gist.github.com/rxaviers/7360908>

## How to add links in markdown?

- [Go to the Support Web Site](#)
- [This is a stupid website](#)
- or you can just paste it in raw text and it will turn to link automatically

```
[Go to the Support Web Site](https://www.baidu.com)
[This is a stupid website](https://www.baidu.com)
```

## C++ learning

---

### 基础篇

```
{
    #include<iostream>
    using namespace std;
    cout<<" a variable "<<a<<endl;
    cin>>a;
    // if 不要加分號 ;
}
```

### 三目運算符

表達式1 ? 表達式2 : 表達式3

```
c=(a>b?a:b);
```

switch 语句//switch判断时只能是整型或字符，不能是区间

case 结果1：执行语句；break；//break 代表退出当前分支，不然会继续执行

case 结果2：执行语句；break；

default：执行语句；break；

rand()%100 //生成0~99的随机数

```
#include<ctime>
srand((unsigned int)time (NULL)); //添加随机数种子，防止每次随机数都一样
```

循环有哪些？

do-while；while；for；

跳转语

break ; continue ; goto ;

数组

```
int arr[5];
int arr[5]={10,20,30,40,50};
如果初始化时没有全部写完会自动补零;
取地址符号&;
sizeof(arr)/sizeof(arr[0])-1;
二维数组 int arr[][]=0;
```

函数

1. 返回值类型
2. 函数名
3. 参数列表
4. 函数体语句
5. return表达式

```
语法：
返回值类型 函数名 参数列表
{
    函数体语句
    return 表达式
}
```

函数调用语法： 函数名称 （ 参数 ）

值传递 把实际参数的值传递给形式参数称为值传递；值传递的过程不影响实际参数；

**void** 表示没有参数返回值  
函数的声明 为了把函数写到main函数的后面

函数的分文件编写：为了让代码更加清晰

1. 创建后缀名为.h的头文件
2. 创建后缀名为.cpp的源文件
3. 在头文件中写函数的声明
4. 在源文件中写函数的定义

指针

作用：可以间接访问内存

指针定义的语法：数据类型\*指针变量名；

```
int a=100;
```

```
int *p;
```

```
p=&a;
```

指针前加\*代表解引用，找到指针指向的内存中的数据

```
*p=1000;
```

```
cout << a << endl;
```

在32位操作系统中指针占4个字节；64位中占有8个字节；

空指针：用来给指针变量进行初始化；

空指针时不可以进行访问的。0~255之间的内存编号时系统占用内存，不允许用户访问。

野指针：指针变量指向非法的内存空间

Const来修饰指针

const修饰指针有三种情况：

1. const修饰指针——常量指针
2. const修饰常量——指针常量
3. const既修饰指针又修饰常量

```
const int * p =&a;
```

常量指针：指针指向可以修改，但指针指向的值不可以改

```
int * const p =&a;
```

指针常量：指针指向的值可以修改，指向不可以改；

```
const int * const p=&a;
```

指针的指向和指向的值都不能修改；

指针和数组：利用指针访问数组中的元素；

数组名就是数组的首地址；

指针和函数：利用指针做函数的参数，可以修改实参中的值

## 结构体

结构体属于用户自定义的数据类型，允许用户存储不同的数据类型

语法：struct 结构体名{结构体成员列表}；

通过结构体创建变量的三种方式：

- struct 结构体名 变量名
- struct 结构体名 变量名={成员1值·成员2值.....}
- 定义结构体时顺便创建变量

```
struct Student
{
    //成员列表
    string name;
    int age;
    int score;
}

#include<string>
struct Student s1
//通过.来访问结构体变量中的属性
s1.name="zhangsan";
s1.age=18;
s1.score=100;
struct Student s2={..}
Student s3;//结构体变量定义时struct关键字不能省略，创建一个新变量时可以省略
```

## 结构体数组

作用：将自定义的结构体放入到数组中方便维护

语法： `struct 结构体名 数组名[元素个数]={ {}, {}, ... {} }`

```
struct Student arr[3]=
{
    {"张三", 18, 100},
    {"李四", 19, 20},
    {"王五", 20, 32}
};

arr[2].name="赵六";
```

## 结构体指针

作用：通过指针访问结构体中的成员

- `struct student *p=&arr;` //创建的是结构体指针
- `p->name;` //结构体指针通过->来访问结构体中的成员

## 结构体嵌套

- `t1.stu.name="zhangsan";`

## 结构体做函数参数

如果不想修改主函数中的数据，用值传递，反之用地址传递；

用const来防止误操作;使得所指的地址只可以读;

将函数中的形参改为指针，可以减少内存空间，而且不会复制新的副本出来;

## C++ 核心编程

---

### 内存分区模型

C++程序执行时，将内存大方向分为4个区域

- 代码区：存放函数体的二进制代码，由操作系统进行管理
- 全局区：存放全局变量和静态变量以及常量
- 栈区：由编译器自动分配释放，存放函数的参数值，局部变量等
- 堆区：由程序员分配和释放，若程序员不释放，程序结束时由操作系统释放
- 意义：不同区域存放的数据，赋予不同的生命周期，使编程更灵活

#### 静态变量

```
static int i=10;
```

全局区中：全局变量，静态变量static，常量-字符串常量和const修饰的全局常量。

栈区数据注意事项：不要返回局部变量的地址。不要返回局部变量的引用。

在C++中主要利用new来开辟数据到堆区。new返回的是堆区的地址；用delete释放内存；

### 引用

作用：给变量起别名

语法：数据类型 &别名=原名

- 引用必须初始化
- 引用在初始化之后，不可以改变
- 引用的本质在C++内部实现是一个指针常量

### 函数提高

#### 函数默认参数

在c++中，函数的形参列表中的形参是可以有默认值的；

#### 函数占位参数

C++中函数的形参列表里可以有占位参数，用来做占位，调用函数时必须填补该位置。

#### 函数重载

- 同一个作用域下
- 函数名称相同
- 函数参数类型不同或者个数不同或者顺序不同

- 注意：函数的返回值不可以作为函数重载的条件

## 类和对象

C++面向对象的三大特性为：封装、继承、多态

### 封装

封装的意义：将属性和行为作为一个整体；将属性和行为加以权限控制

语法：class 类名 {访问权限：属性/行为};

类中的属性和行为统称为成员；

属性 又称为 成员属性 成员变量

行为 又称为 成员函数 成员方法

类在设计时，可以把属性和行为放在不同的权限下，加以控制

1. public 公共权限 成员 类内可以访问，类外也可以访问
2. protected 保护权限 成员 类内可以访问，类外不可以访问 儿子可以访问父亲中的保护内容
3. private 私有权限 成员 类内可以访问，类外不可以访问 儿子不可以访问父亲中的私有内容

### struct 和 class 的区别

唯一的区别就在于默认访问权限不同

- struct 默认权限为公共
- class 默认权限为私有

### 成员属性设为私有

1. 将所有成员属性设为私有，可以自己控制读写权限
2. 对于写权限，我们可以检测数据的有效性

在类中可以让另一个类作为本类中的成员

### 对象的初始化和清理

出厂设置；对象销毁前清理数据的设置。

- 构造函数和析构函数；系统自动调用，为成员属性赋值。

函数名和类名相同

构造函数的语法：类名 ( ) {}

构造函数实现初始化

析构函数的语法: ~类名 ( ) {}

析构函数实现清理

### 构造函数的分类和调用

- 两种分类方式：
  - 按参数分为：有参构造和无参构造
  - 按类型分为：普通构造和拷贝构造



- 三种调用方式：

- 括号法
- 显示法
- 隐式转换法

Person p4=10; 相当于 Person p4 = Person(10)

匿名对象 person(10)

不要用拷贝构造函数 初始化匿名对象;

### 拷贝构造函数调用时机

C++中拷贝构造函数调用时机通常有三种情况

- 使用一个已创建完的对象来初始化一个新对象
- 值传递方式给参数传值
- 以值方式返回局部变量

### 构造函数调用规则

默认情况下，c++编译器至少给一个类添加3个函数

1. 默认构造函数（无参，函数体为空）
2. 默认析构函数（无参，函数体为空）
3. 默认拷贝函数，对属性进行值拷贝

构造函数调用规则如下：

- 如果用户定义有参构造函数，c++不在提供默认无参构造，但是会提供默认拷贝构造
- 如果用户定义拷贝构造函数，c++不会再提供其他构造函数