# Introduction to C++ (Season 1)

## Unit 3: More than C

## 第3单元:更上一层楼 – 超越C的语法

### Section 4 : Simplified Memory Model for C/C++

### 第4节： C/C++的简化内存模型

# Simplified Memory Model (C++的内存模型)

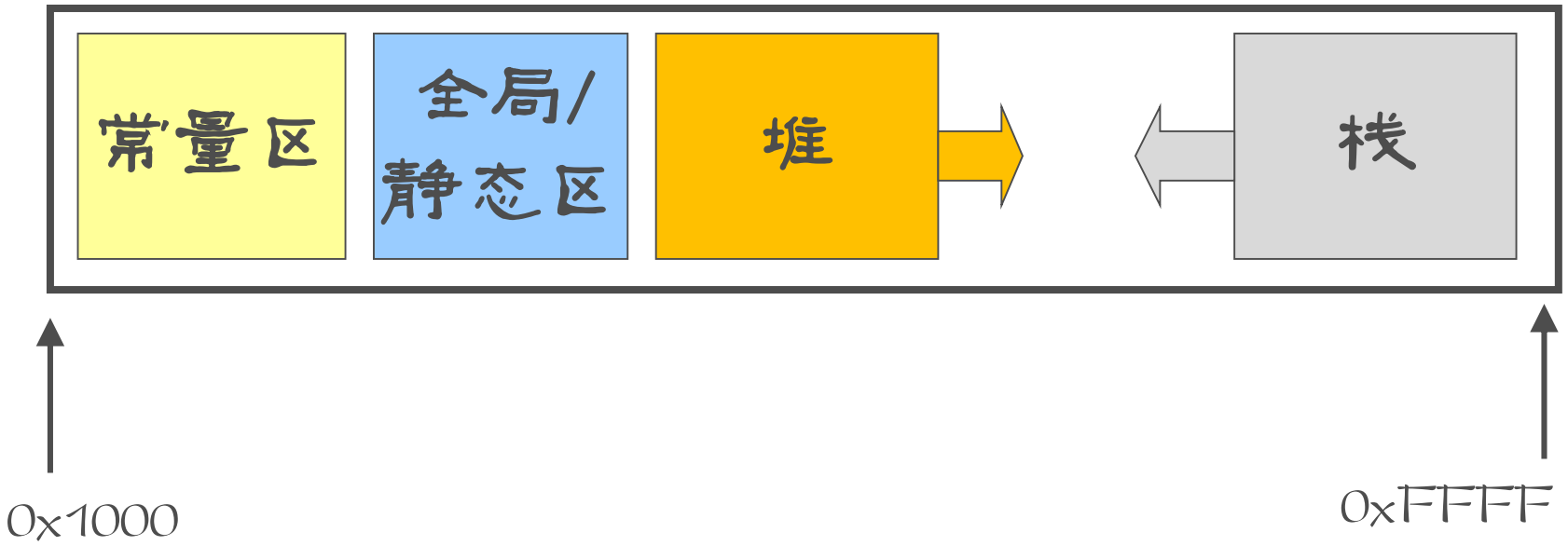
注意:
这个简化模型仅用于初学者示意
与实际模型并不完全一致

1. Stack (栈)
   - 编译器自动分配释放
2. Heap (堆)
   - 一般由程序员分配释放，若程序员不释放，程序结束时可能由OS回收
3. Global/Static (全局区/静态区)
   - 全局变量和静态变量的存储是放在一块的。
   - 可以简单认为：
     - 程序启动全局/静态变量就在此处
     - 程序结束释放
4. Constant (常量区)
   - 可以简单理解为所有常量都放在一起
   - 该区域内容不可修改

# Example of memory allocation (C++程序的内存示例)

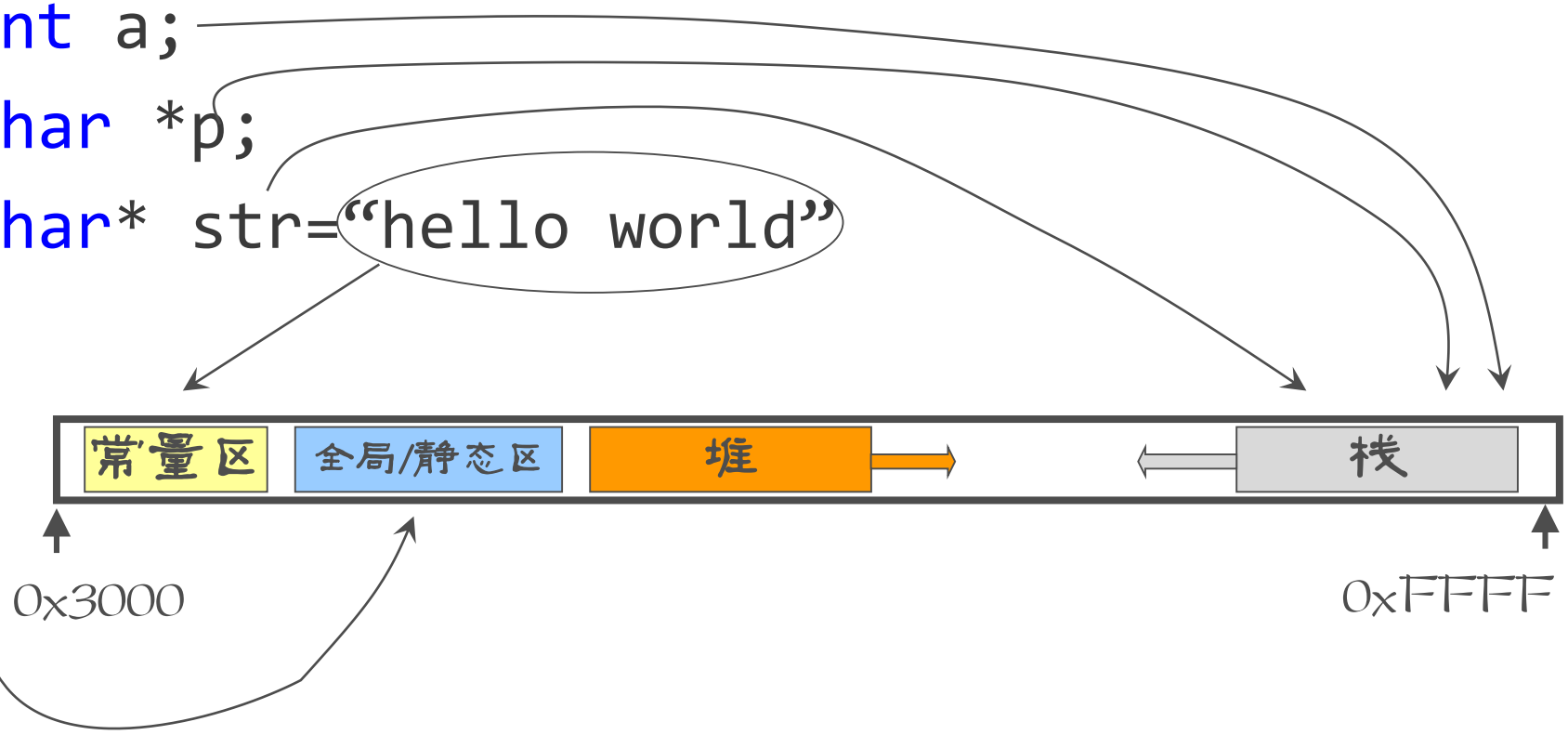
- 堆向高地址方向生长
- 栈向低地址方向生长

**注意:**
这个示例模型仅用于初学者示意
与实际并不完全一致



# Location of a variable (变量存放位置)

```
int arr[3];
int myFunc()
{
    int a;
    char *p;
    char* str="hello world"
}
```



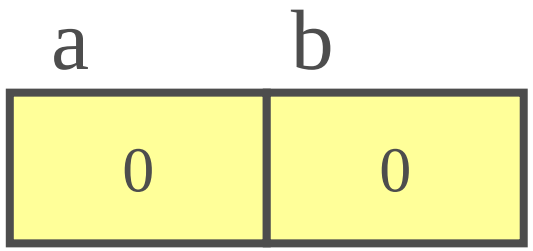
# Location of a variable (变量存放位置) – 续

```
int myFunc1(int* pi)
{
   char *pc;
   pc= static_cast<char*>  new char[8];
}
```

# 普通变量内存模型

- 普通变量

```
int a, b=0;    a=b;
```




- a和b都是变量的名，对a和b的访问实际上访问的是a和b这两个变量中存储的值
- a和b的地址分别是 &a 和 &b

# 数组内存模型

- 对于数组a[]，a是数组a[]的首地址的别名
- 要访问每个数组元素的值，使用a[0], a[1],…
- 要访问一个地址所存的内容，使用“*”
  - 访问数组中第一个元素可以使用 *(a+0)
  - 访问数组中第二个元素 *(a+1)