X 语言程序设计

本节内容

结构体的定义与初始化

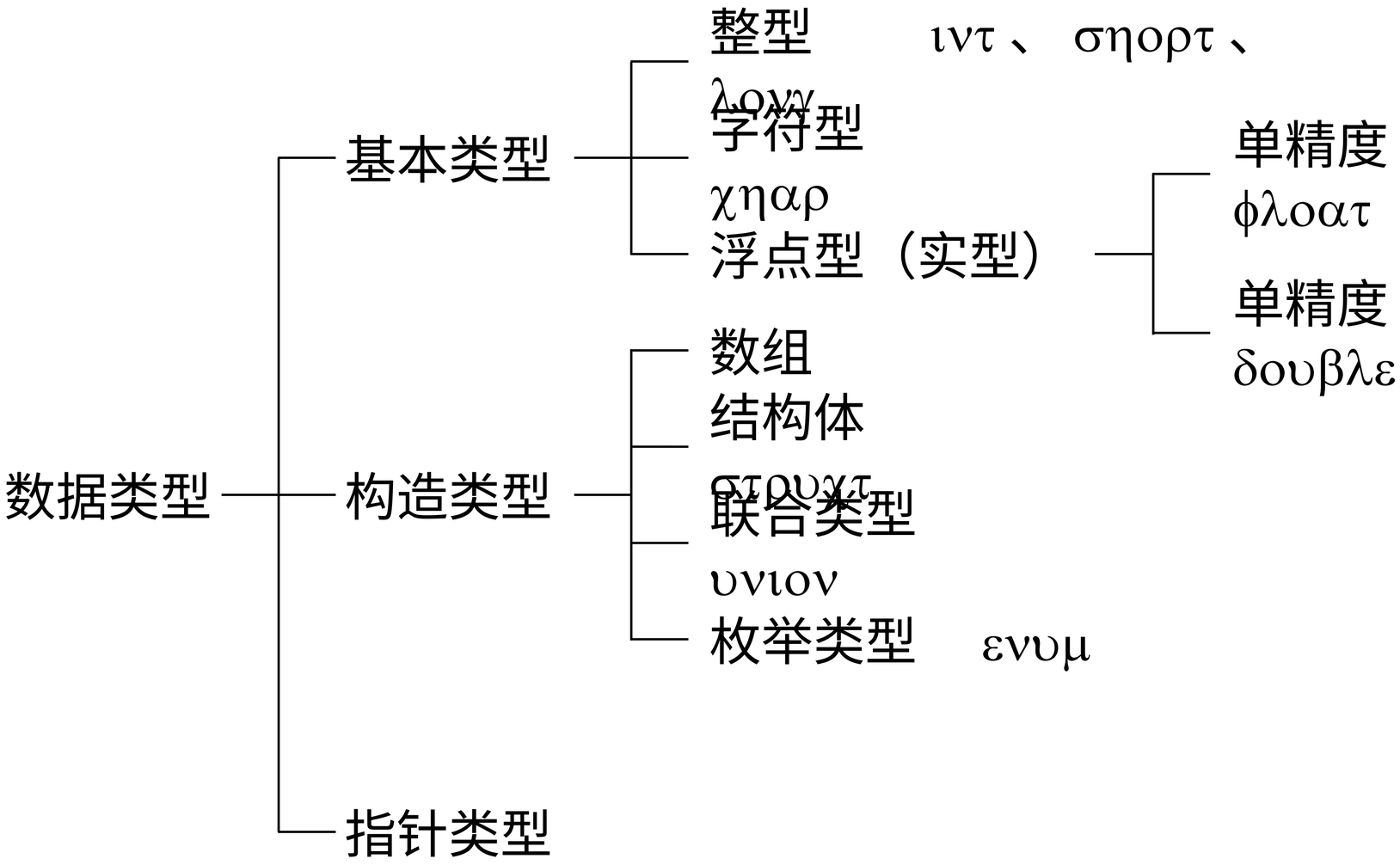
结构体与指针

内存对齐与内存补齐

τψπεδεϕ

枚举

数据类型


数据类型
基本类型
整型
ιντ、 σηορτ、 λονγ
字符型
χηαρ
浮点型（实型）
单精度
φλοατ
单精度
δουβλε
构造类型
数组
结构体
στρυχτ
联合类型
υνιον
枚举类型
ενυμ
指针类型

结构体的定义

结构体是一个或多个变量的集合，这些变量可以是不同的类型。

声名语法：

```
struct  结构体名
{
    数据类型  变量名；
    数据类型  变量名；
    ……
};
```

```
struct point
{
    int x;
    int y;
};
```

结构体变量的声明与初始化

```
struct 结构体名 变量名;
```

```
struct point p;
```

```
struct 结构体名 变量名 = {值1, 值2 ……};
```

```
struct point p = {1, 2};
```

结构体变量的声明与初始化

```
struct 结构体名 变量名 = {.变量名=值, .变量名=值 ……};
```

```
struct point p = {.x=1, .y=2};
```

```
struct point p = {.y=2, .x=1}
```

结构体变量赋值

变量名．结构体内部变量名　=　值

```
p.x = 10;
p.y = 15
```

结构体与指针

在一些场景中，如果传递给函数的结构体很大，使用指针方式的效率通常更高。

πp 指向一个 πoιντ 结构体

```
struct point *pp;
```

为 πp 所指向的结构体中的属性赋值

```
(*pp).x = 10;
(*pp).y = 5;
```

为了使用方便， X 语言提供了另外一种简写方式

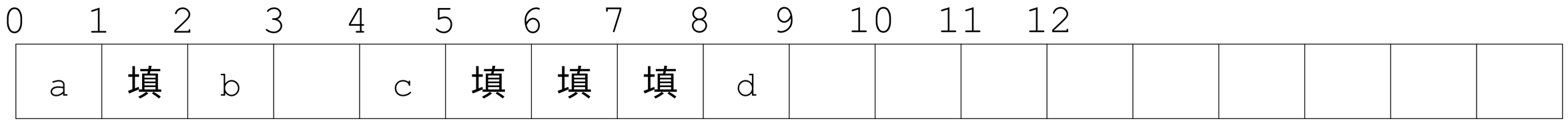
```
pp->x = 10;
pp->y = 5;
```

案例：结构体 / τεστ3.χ

内存对齐

结构体中每个成员的起始地址，必须是其自身大小的整数倍，为此可能需要在中间填充若干字节

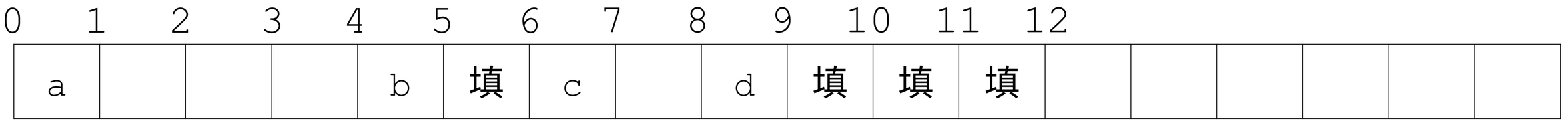
```
struct x
{
    char a; //1 字节
    short b; //2 字节
    char c;//1 字节
    int d;//4 字节
};
```



内存补齐

结构体的总字节数，必须是其最大成员大小的整数倍，为此可能需要在末尾填充若干字节

```
struct y
{
    int a; //4 字节
    char b; //1 字节
    short c;//2 字节
    char d;//1 字节
};
```



类型定义

```
typedef  数据类型  别名
```

```
typedef int zx
```

案例：结构体 /
τεστ4.χ

类型定义

```c
struct point
{
    int x;
    int y;
};

int main(int argc, char const *argv[])
{
    struct point p;
    p.x = 5;
    p.y = 10;
    printf("%d\n", p.x);
    printf("%d\n", p.y);
    return 0;
}
```

每次声明结构体变量都要写 στρυχτ 关键字，很麻烦，而且逻辑上也很难受，τψπεδεφ 可以解决这个问题。

类型定义

```
typedef struct  结构体名
{
    数据类型  变量名；
    数据类型  变量名；
    ……
} 别名；
```

```
typedef struct
{
    数据类型  变量名；
    数据类型  变量名；
    ……
} 别名；
```

类型定义

```
typedef struct point
{
    int x;
    int y;
}po;
```

```
typedef struct
{
    int x;
    int y;
}po;
```

枚举

将变量的值一一列举出来，变量的值只限于列举出来的值的范围内

在枚举中列出的每一个值，称为枚举元素
每一个枚举元素由系统定义了一个用序号表示的数值，从 0 开始，分别为 0 ， 1 ， 2 …

语法：

```
enum  枚举名
{
      枚举元素……
}
```