命令行参数

```
// argc: 命令行参数的总个数
// argv: 命令行参数所组成的数组
// envp: 环境变量所组成的数组，以NULL结尾
int main(int argc, char *argv[], char *envp[])
{
    ...
}
```

结构体

定义

```
struct 结构体名 {
    类型名1 成员名1;
    类型名2 成员名2;
    ...
};

// 例如
struct student_info {
    char name[20];
    int age;
    double score;
};
// 定义studenet_info结构体变量stu1并初始化
struct student_info stu1 = { "Jack", 2, 100.0 };
```

访问

```
...
printf("%s\n", stu1.name);          // 打印name成员的值
printf("%d\n", stu1.age);           // 打印age成员的值
printf("%lf\n", stu1.score);        // 打印score成员的值

// 等价于memcpy(&stu2, &stu1, sizeof(struct student_info));
// 属于浅拷贝
struct student_info stu2 = stu1;
```

原理：

```
struct type s;
s.member;   // *(member type *)((int)&s + member offset)
```

对齐

在项目设置中 —> **C/C++** —> **代码生成** —>**结构体成员对齐**选项

布局

计算规则：

- 设编译选项设定的对齐值为 `Zp`
- 每个结构体成员的偏移量为 `Offset`
- 必须满足：`Offset % min(sizeof(member type), Zp) == 0`
    - 如果是嵌套结构体：`Offset % min(stru_ali, Zp) == 0`

大小

计算规则：

- 设整个结构体自身的对齐值为`stru_ali`
- 成员中最大的成员的大小：`stru__ali = max(sizeof(member1 type1), ..., sizeof(membern type))`
- `stru_ali = min(stru_ali, Zp)`
- `sizeof(struct xxx) % stru_ali == 0`

offsetof

取成员偏移量

```
define offsetof(s, m) (size_t)(&(((s *)NULL)->m))
```