C++ 指针与引用学习文档

目录

---

1. 指针与引用的基本概念

1.1 指针

- **定义**：指针是一个变量，其值为另一个变量的地址。
- **作用**：通过指针可以间接访问和操作目标变量的值。
- **语法**：

```
int a = 10;
int *p = &a; // 指针 p 存储了变量 a 的地址
*p = 20;     // 通过解引用修改 a 的值
```

1.2 引用

- **定义**：引用是变量的别名，用于直接访问目标变量。
- **特性**：
  1. 引用在声明时必须初始化。
  2. 引用一旦绑定到一个变量，就不可更改绑定。
- **语法**：

```
int a = 10;
int &ref = a; // ref 是 a 的引用
ref = 20;     // 修改 ref 会改变 a 的值
```

---

2. const 与指针、引用的结合使用

2.1 const 与变量

- **const 的作用**：修饰变量，表示该变量的值不可修改。

- **示例**：

```
const int a = 10; // a 是常量，值不可修改
```

2.2 const 与指针

- **指针与 const 的结合有以下几种情况**：

1. **指针指向的值不可修改**：

```
const int *p = &a; // p 指向 const int 类型，不能通过 p 修改 a 的值
```

2. **指针本身不可修改**：

```
int *const p = &a; // p 是常量指针，指向的地址不可修改
```

3. **两者都不可修改**：

```
const int *const p = &a; // p 的地址和值均不可修改
```

2.3 const 与引用

- **引用与 const 的结合主要用于指向 const 类型的变量或字面值**：

```
const int &ref = 20; // 合法，允许引用绑定到临时变量
```

---

3. 强制类型转换与内存操作

3.1 强制类型转换

- **定义**：允许将一种类型的变量强制转换为另一种类型。
- **语法**：

```
int *p = (int*)0x0018ff44; // 将地址 0x0018ff44 强制转换为 int 指针
```

3.2 内存操作的风险

- **注意事项**：
    1. 确保目标地址在当前程序的权限范围内。
    2. 若访问非法地址，可能引发未定义行为或程序崩溃（如段错误）。

---

4. 示例代码与解析

4.1 示例 1：引用绑定临时变量

```
const int &a = 20;
```

- **解析**：允许 `const` 引用绑定到字面值（临时变量），编译器会为字面值生成一个临时变量，并将引用绑定到此临时变量。

4.2 示例 2：指针与引用结合

```
int a = 10;
int *p = &a;
int *&q = p; // 合法
```

- **解析**：`q` 是一个指针引用，可以绑定到普通指针 `p`，通过 `q` 可以修改 `p` 的值。

4.3 示例 3：常见错误

代码：

```
int a = 10;
int *const p = &a;
int *&q = p; // 错误
```

原因：

- `p` 是常量指针（指针值不可修改），而 `q` 是指针引用（要求引用非常量指针）。
- 类型不匹配，编译器报错。

4.4 示例 4：操作内存地址

代码：

```
int *p = (int*)0x0018ff44;
*p = 10; // 写入值 10
```

- **解析**：将地址 `0x0018ff44` 强制转换为指针后，写入值 `10`。需确保地址合法，否则会引发错误。

---

5. 常见错误与调试技巧

5.1 类型不匹配

```
const int *p = &a;
int *&q = p; // 错误
```

- **原因**：`q` 是 `int*` 引用，要求绑定到普通指针，而 `p` 是 `const int*`。

5.2 非法内存访问

```
int *p = (int*)0x0018ff44;
*p = 10; // 若地址无效，将导致段错误
```

- **解决方法**：检查地址合法性，避免非法操作。

---

6. 练习题与答案

题目 1

```
int a = 10;
int *p = &a;
const int *&q = p; // 是否合法?
```

答案：不合法。`p` 是 `int*` 类型，与 `const int*&` 类型不兼容。

题目 2

```
int a = 10;
const int *p = &a;
int *&q = p; // 是否合法?
```

答案：不合法。类型不匹配，`q` 需要 `int*` 类型，而 `p` 是 `const int*`。

7. 总结与扩展

7.1 关键点

1. **指针与引用**：
    - 指针存储地址，可解引用访问目标变量。
    - 引用是变量的别名，绑定后不可修改。
2. `const` **的使用**：
    - 使用 `const` 限制指针或引用的修改权限，增强代码安全性。
3. **强制类型转换与风险**：
    - 确保地址合法，避免非法内存访问。