C++

笔记本： 我的第一个笔记本
创建时间： 2020/9/14 19:01 **更新时间：** 2020/9/14 22:08
作者： 1272209351@qq.com

C++

C语言之父-----里奇
C++98 里边有63个关键字

asm	do	if	return	try	continue
auto	double	inline	short	typedef	for
bool	dynamic_cast	int	signed	typeid	public
break	else	long	sizeof	typename	throw
case	enum	mutable	static	union	wchar_t
catch	explicit	namespace	static_cast	unsigned	default
char	export	new	struct	using	friend
class	extern	operator	switch	virtual	register
const	false	private	template	void	true
const_cast	float	protected	this	volatile	while
delete	goto	reinterpret_cast			

一、命名空间的三种方式：
1、常用方式；2、命名空间可以嵌套；3、同一个工程中可以定义多个名字相同的命名空间

```cpp
namespace N1
{
    int a = 10;
    int b = 20;

    int Add(int left, int right)
    {
        return left + right;
    }
}
```

```cpp
// 2. 命名空间可以嵌套
namespace N2
{
    int a = 10;
    int b = 20;

    int Sub(int left, int right)
    {
        return left - right;
    }

    namespace N3
    {
        int c = 10;
        int d = 20;

        int Mul(int left, int right)
        {
            return left*right;
        }
    }
```

```cpp
// 3. 在同一工程中，可以定义多个名字相同的命名空间
// 不会冲突
// 编译器会将多个相同名称的命名空间合并成一个
namespace N1
{
    int Div(int left, int right)
    {
        return left / right;
    }
}
```

相同作用域中不能出现相同的变量名字
相同名称的多个命名空间中：也不能出现相同的名字 因为编译器会将多个相同名称的命名空间最终合并成一个

```cpp
namespace N
{
    int a = 10;

    //嵌套的方式----该种方式不会冲突
    // 相当于外层N命名空间中又包含了一个N的命名空间
    namespace N
    {
        int b = 10;
        int a = 10;
    }
}


int main()
{

    return 0;
}
```

命名空间的访问方式：

：： 作用域运算符

```
namespace N
{
    int a = 10;
    int b = 20;

    int Add(int left, int right)
    {
        return left + right;
    }
}

int a = 20;

int main()
{
    int a = 30;

    printf("%d\n", a);
    return 0;
}
```

```
int main()
{
    int a = 30;

    // 就近原则
    printf("%d\n", a);

    // 如果访问全局作用域中的a
    // ::作用域运算符
    // ::a  明确说明要访问全局作用域中的a
    printf("%d\n", ::a);
    return 0;
}
```

```
// 访问N命名空间中的a
printf("%d\n", N::a);
```

```
// 该场景：对N命名命名空间中某些成员访问的非常频繁

using N::a;

int main()
{
    // 访问N命名空间中的a
    printf("%d\n", N::a);
    printf("%d\n", N::a);
    printf("%d\n", N::a);
    printf("%d\n", N::a);
    printf("%d\n", N::a);
    printf("%d\n", N::a);
    printf("%d\n", N::a);

    // 为了写代码简单，想要直接访问N命名空间中的a
    printf("%d\n", a);
    return 0;
}
```

using N : : a
该语句加上之后相当于将N 命名空间之内的 a 当成当前文件的一个全局变量来使用

```
// 该场景：对N命名命名空间中某些成员访问的非常频繁
// 优点：写代码简单了
// 缺点：如果该文件中有相同名称的变量或者函数，就会产生冲突
// 如果产生冲突，怎么办？----按照方式1使用即可

// 该条语句加上之后，相当于将N命名空间中的a当成当前文件的一个全局变量来使用
using N::a;


// 如果该文件中也有一个a，必然会产生冲突，只能按照方式1来进行使用
//int a = 10;

int main()
{
    int a = 10;
```

C++中的输入和输出：
C语言的输入输出方式在C++中依旧可以使用-----兼容

```
#pragma  warning (disable:4996)
#include<iostream>
using namespace std ;

cin >> endl;
cout << endl;
```