§.typedef声明新类型

1.含义

用新的名称来等价代替已有的数据类型

★ 不产生新类型，仅使原有类型有新的名称

★ 建议声明的新类型名称为大写，与系统类型区分

2.使用

声明名称：

```
typedef 已有类型  新名称;
typedef int INTEGER;
typedef struct student STUDENT; //C++无此需求
typedef int ARRAY[10];
typedef char * STRING;
```

```
struct student {
    ...
};
struct student s1;//C方式
STUDENT s1;//C方式
student s1;//C++方式
```

C方式的另一种小技巧，将无名结构体声明为student

```
typedef struct {
    ...
} student;
student s1;//C方式
```

用新名称定义变量及等价对应关系：

INTEGER i,j;		int i,j;
STUDENT s1,s2[10],*s3;		student s1,s2[10],*s3;
ARRAY a,b[5];	等价方式	int a[10],b[5][10];
STRING p,x[10];		char *p,*x[10];

§.typedef声明新类型

特别说明：
★ typedef声明新类型不属于必须使用的方法
★ 请大家自行评估可读性并选择适合自己的风格，不强求

3.声明新类型的一般步骤
① 以现有类型定义一个变量
② 将变量名替换为新类型名
③ 加typedef
④ 完成,可定义新类型的变量

```
① int i;
② int INTEGER;
③ typedef int INTEGER;
④ INTEGER i,j;
```

```
① int a[10];
② int ARRAY[10];
③ typedef int ARRAY[10];
④ ARRAY a,b[5];
```

```
① char *s;
② char *STRING;
③ typedef char *STRING;
④ STRING p,x[10];
```

```
#include <iostream>
#include <cstring>
using namespace std;

typedef const char* STRING;

int main()
{
    const char *p1="house";
    STRING p2="horse";

    if (strcmp(p1, p2)>0)
        cout<<"大于"<<endl;
    else
        cout<<"不大于"<<endl;

    return 0;
}
```

Microsoft
大于

★ 使用方法与原来的类型一致，与原类型可直接混用，不需要进行强制类型转换
★ 使用重载函数时，若参数类型是由typedef声明的不同名称的相同类型，则会产生二义性