内联函数

- 内联函数具有普通函数的所有行为，唯一不同之处在于内联函数会在适当的地方像预定义宏一样展开，所以不需要函数调用的开销。
- 在普通函数(非成员函数)函数前面加上inline关键字使之成为内联函数。但是必须注意必须函数体和声明结合在一起，否则编译器将它作为普通函数来对待。
- 内联函数相较于普通函数的优势是省去了函数调用时的压栈，跳转，返回的开销，但是占用空间。
- 内联仅仅只是给编译器一个建议，编译器不一定会接受这种建议，如果你没有将函数声明为内联函数，那么编译器也可能将此函数做内联编译。一个好的编译器将会内联小的、简单的函数。

内联函数和宏定义区别

```
7  inline int sum(int a,int b){
8      return a<b?a:b;
9  }
10 int main() {
11
12     int a=1,b=3;
13     int c=SUM(++a,b);    // 宏替换展开 (++a)<(b)?(++a):(b)
14     printf("a=%d,b=%d,c=%d\n", a,b,c);
15
16     a=1,b=3;
17     c=sum(++a,b);
18     printf("a=%d,b=%d,c=%d\n", a,b,c);
19
20     return 0;
21 }
```

快读法

```
inline int read() {
    int x = 0;
    int f = 1;
    char ch = getchar();
    while (ch<'0' || ch>'9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }

    while (ch>='0' && ch<='9') {
        x = x * 10 + ch - '0';
        ch = getchar();
    }

    return x * f;
}
```

```
/*
6
1 2 3 4 5 6
*/

int n, a[101];
int main() {
    n=read();
    For(i,1,n)
        a[i]=read();
    For(i,1,n)
        cout << a[i] << " ";

    return 0;
}
```


函数重载

在传统c语言中，函数名必须是唯一的，程序中不允许出现同名的函数。在c++中是允许出现同名的函数，这种现象称为函数重载。

函数重载的目的就是为了方便的使用函数名。

- 同一个作用域
- 参数个数不同
- 参数类型不同
- 参数顺序不同