快速读入(出)/读入(出)优化(模板)

一、前言

众所周知: scanf 比 cin 快得多， printf 比 cout 快得多。这里的优化思想就是运用 getchar(putchar) 比 scanf(printf) 快的优势写一个模板，注意，这里只是针对整数，浮点数目前不支持。

二、输入

1.入门级的cin

```
cin >> a >> b;
```

2.普遍的scanf

```
scanf("%d%d", &a, &b);
```

3.关闭流同步的cin

```
ios::sync_with_stdio(false);
cin >> a >> b;
```

4.读入优化read

(1).读入优化 read ，原理是将数字按照数位一个个读进来(比如数字123)就会依次读入 1, 2, 3，每次将数字乘10再加当前数字)

```
int read() {//'&'表示引用，也就是说x是一个实参，在函数中改变了x的值就意味着在外面x的值也会被改变
    int f = 1;//标记正负
    int x = 0;//结果
    char s = getchar();//读入第一个字符
    while (!isdigit(s)) {//不是数字
        if (s == '-') //不能直接把f = -1，有可能输入的不是'-'而是其它的东西
            f = -1;
        s = getchar(); //继续读
    }
    while (isdigit(s)) {//是数字(一旦不是数字就意味着输入结束了)
        x = x * 10 + s - '0';
        s = getchar();
    }
    return x * f; //改变正负
}
int main() {
    a = read();
}
```

(2).上面这个只能读入int，其它类型的不能读入，故流出了下面这个版本(代码的 template <... 是为了能同时读入多种类型变量 (unsigned)int ， longlong ， short 都行):

```
template <typename _Tp>
inline _Tp read(_Tp &x) {
    char ch = getchar(), sgn = 0; x = 0;
    while (ch ^ '-' && !isdigit(ch)) ch = getchar();
    if (ch == '-') ch = getchar(), sgn = 1;
    while (isdigit(ch)) x = x * 10 + ch - '0', ch = getchar();
    if (sgn) x = -x;
    return x;
```

```
}
int main() {
    b = read(a);
}
```

5.读入优化fread

准备区域赛时发现 fread 表现更优秀(这里和上头差不多，只不过是一次性读入一大串字符， gc() 函数每调用一次返回一个字符，相当于上面的 getchar()).

```
struct ios_in {
    inline char gc() {
        static char buf[MAXN], *l, *r;
        return (l == r) && (r = (l = buf) + fread(buf, 1, MAXN, stdin), l == r) ? EOF : *l++;
    }
    template <typename _Tp>
    inline ios_in & operator >> (_Tp &x) {
        static char ch, sgn;
        for (sgn = 0, ch = gc(); !isdigit(ch); ch = gc()) {
            if (!~ch) return *this;
            sgn |= ch == '-';
        }
        for (x = 0; isdigit(ch); ch = gc())
            x = (x << 1) + (x << 3) + (ch ^ '0');
        sgn && (x = -x);
        return *this;
    }
}Cin;
int main() {
    Cin >> a;
}
```

比较

C/C++几种输入方法的速度大致为: c i n < < s c a n f < c i n (关 闭 流 同 步) < r e a d < < f r e a d cin<<scanf<cin(关闭流同步)<read<<fread cin<<scanf<cin(关闭流同步)<read<<fread.
三、输出

1.入门级的cout

```
cout << a << b;
```

2.普遍的printf

```
printf("%d%d", a, b);
```

3.关闭流同步的cout

```
ios::sync_with_stdio(false);
cout << a << b;
```

4.读出优化write

```
void write(int x) {
     if (x < 0) putchar('-'), x = -x;
     if (x > 9) write(x / 10);
     putchar(x % 10 + '0');
}
int main() {
    write(a);
}
```

5.数组优化

```
struct ios_out {
    template <typename _Tp>
    inline void operator << (_Tp &x) {
        char F[MAXN];
        _Tp tmp = x > 0 ? x : (putchar('-'), -x);
        int cnt = 0;
        while (tmp) {
            F[cnt++] = tmp % 10 + '0';
            tmp /= 10;
        }
        while (cnt) putchar(F[--cnt]);
    }
}Cout;
int main() {
    Cout << a;
}
```

比较

C/C++几种输入方法的速度大致为: c o u t < < p r i n t f < c o u t (关 闭 流 同 步) < w r i t e < 数 组 优 化 cout<<printf<cout(关闭流同步)<write<数组优化 cout<<printf<cout(关闭流同步)<write<数组优化.