پیشگیری بهتر از درمان

حقه‌های برنامه‌نویسی

میلاد پیکانی

حقه‌های برنامه‌نویسی، به‌روش‌هایی گفته می‌شود که به‌کمک آنها بتوان کاری کرد که میزان خطاهایی که ممکن است در طول برنامه‌نویسی رخ دهند، به‌حد زیادی کاهش پیدا کند. برخی از این حقه‌ها به‌عنوان «برنامه‌نویسی تدافعی» شناخته می‌شوند. برنامه‌نویسی تدافعی جنبه‌های مختلفی دارد. از جنبه‌های امنیتی یک نرم‌افزار گرفته تا این‌که یک کد تمیز، سالم و بدون خطا باشد.

پیش از این‌که چیزی را بخواهیم شروع کنیم، لطفا هیچ پیش‌فرضی در مورد کامپایلر نداشته باشید. کامپایلرها از ویژگی‌های مختلفی پشتیبانی می‌کنند که ممکن است در همه آنها ثابت نباشد. مثلا این تکه کد را نگاه کنید:

```
Array[i] = i++;
```

نتیجه عبارت بالا چه خواهد شد؟ آیا اول عملیات جایگزینی انجام می‌شود و بعد یک واحد به i اضافه می‌شود یا برعکس این موضوع صادق خواهد بود؟ این مساله تا حدودی گنگ است، اما یک برنامه‌نویس خوب همواره از نوشتن کدهایی که ابهام دارد، خودداری می‌کند. هیچ وقت فراموش نکنید که نوشتن یک کد تمیز، خوانا و قابل ردیابی، بسیار بهتر از یک کد کوتاه، اما غیرقابل ردیابی است که نیاز به هوش بالایی دارد.

۱- عبارت اگر برای مقایسه ثابت‌ها

وقتی می‌خواهید مقدار ثابتی را با یک متغیر مقایسه کنید، به دو روش برمی‌خورید. نخستین حالت آن در این شرط مقایسه‌ای، چیزی شبیه به کد زیر خواهد بود:

```
if (input_temp == CONST_VAL ) {
                //....some code goes here
}
```

و حالت دوم به‌صورت زیر:

```
if (CONST_VAL == input_temp ) {
                ///....some code goes here
}
```

حالا به‌نظرتان کدام یک بهتر است؟ روش دوم کمی عجیب به‌نظر می‌رسد، اما بهتر است از آن استفاده شود! چرا؟ این حالت را در نظر بگیرید که به‌طور اشتباهی یک مساوی فراموش شود:

```
if (input_temp = CONST_VAL) {
```

حالا می‌توانید اشتباه کد بالا را درک کنید. در این حالت، همیشه جواب شرط، مقدار درست خواهد بود. این اشتباه وقتی که ثابت را اول بنویسیم برطرف خواهد شد، چرا که کامپایلر خطا می‌دهد.

۲- تقدم عملگرها

اگر بیشتر از یک عملیات وجود داشته باشد، حتما از پرانتز استفاده کنید. این کد را ببینید:

```
var1=2;
var2=10;
var3=4;
temp = var1+var2*var3;
```

اگر منتظرید که عبارت بالا نتیجه ۴۸ را بدهد به حقیقت فکر کنید که نتیجه ۴۲ را خواهید گرفت. اشتباه از کجاست؟

اگر ندانید که عملگر ضرب نسبت به عملگر جمع اولویت بیشتری دارد، در این صورت ایرادی منطقی خواهید داشت که به‌سادگی قابل رفع نخواهد بود. برای سادگی کار می‌توان کد در دو خط را نوشت. البته پرانتزگذاری بهترین راه حل است.

۳- بررسی کد بازگشتی توابع استاندارد

صرف نظر کردن از کدهایی که توابع استاندارد کتابخانه‌ای پس می‌فرستند، تبدیل به امری عادی شده است.

اما حواستان باشد که اگر بخواهید فرض بگیرید که این توابع همواره مقدار درست را برمی‌گردانند، از همین جا ضربه خواهید خورد. به‌عنوان مثال، تابع malloc برای تخصیص پویای حافظه استفاده می‌شود که شکل کلی آن به این صورت است:

```
void *malloc(size_t size);
```

malloc اشاره‌گری را به حافظه‌تازه تخصیص داده شده‌ای برمی‌گرداند که اندازه‌آن را با مقدار size به‌آن داده‌ایم. تکه کد زیر را در نظر بگیرید:

```
int *stk_ptr = malloc(100 * sizeof (int));
if (NULL == stk_ptr) {
// Memory could not be allocated
// Take corrective action
}
```

اگر malloc به هر دلیلی نتوانست حافظه مناسب را تخصیص بدهد، در این صورت اگر بررسی‌ای صورت نگرفته باشد، برنامه‌تان به‌سادگی از کار می‌افتد و شما به‌هیچ‌عنوان نخواهید توانست آن را رهگیری کنید.

۴- غلبه بر محدودیت آرایه

وقتی یک آرایه ایجاد می‌کنید، شاید بزرگترین مشکل‌مان با آن، محدودیت‌اش باشد. کد زیر را در نظر بگیرید:

```
unsinged char input_buffer[10];
unsigned char i;
for (i = 0; i <=10; i++) {
input_buffer[i] = in_port();
}
```

خب در کد بالا انتظار می‌رود که ۱۰بایت داده از ورودی دریافت گردد و داخل بافری که تعریف کرده‌ایم، قرار گیرد. اما به‌خاطر این‌که داخل حلقه for افتاده‌ایم، و همچنین، خود اندیس ۱۰ را هم بررسی می‌کنیم، بنابراین ۱۱بایت از ورودی دریافت می‌شود و داخل آرایه ۱۰ عنصری قرار نمی‌گیرد. از این دست اشکالات باعث می‌شود وقت زیادی را صرف رفع آن کنید.

چندین ابزار هست که می‌توانید به کمک آنها این ایرادها را کشف و تصحیح کنید، PC-Lint و Valgrind از این دست نرم‌افزارها است.

۵- داده از نوع منطقی

به‌طور معمول، نوع داده منطقی (بولین) شامل دو ارزش می‌شود: درست یا نادرست، انتخاب یا خالی، بالا یا پایین، فعال یا غیرفعال، روشن یا خاموش و مانندآن.

بیایید فرض بگیریم که می‌خواهید ثابت‌هایی تعریف کنید که نشان دهنده فعال و غیر فعال باشند. دو روش برای این کار وجود دارد:

```
#define DISABLE 0
#define ENABLE (!DISABLE)
```

که در این دو دستور، فعال مخالف غیرفعال است و عدد ۱ درآن قرار می‌گیرد. و روش دوم:

```
#define DISABLE 0
#define ENABLE 1
```

که به‌طور دستی مقدار ۰ و ۱ را به غیرفعال و فعال بدهیم. در این صورت، اگر بخواهیم حالت‌ها را تغییر بدهیم، در روش اول، کافیست غیرفعال را از ۰ به ۱ تغییر دهیم، تمام کدمان تغییر می‌کند و به‌شیوه جدید فعال می‌شود. اما اگر بخواهیم به‌روش دوم این کار را انجام بدهیم، باید هر دو دستور را تغییر داده و این یعنی بالارفتن احتمال اشتباه.

۶- همیشه از آکولاد استفاده کنید

برخی از برنامه‌نویسان برای دستورات if و حلقه‌های for و while، در زمانی که تنها یک دستور دارند از آکولاد استفاده نمی‌کنند، مثلا عبارت زیر را ببینید:

```
if (SET == timer_is_flag)
    timer_is_flag = CLEAR;
```

یا دستور زیر:

```
for (i = 0; i < BUFF_SIZE; i++)
    buff[i]=i;
```

خب این دستورات به‌خودی خود هیچ اشکالی ندارند، اما اگر به دستورات بالایی، یک دستور اضافه کنید و حواس‌تان نباشد که آکولاد بگذارید، با مشکل بزرگی روبرو می‌شوید، برنامه‌تان مطابق انتظار رفتار نمی‌کند و وقت قابل توجهی را باید صرف کنید تا این را پیدا کنید.

از این‌رو، پیشنهاد می‌شود که با دقت پس از هر دستور شرطی و حلقه، آکولاد بگذارید و سپس اقدام به‌کدنویسی داخل محتوای آن بکنید. در این صورت، پس از چندین روز تمرین، دست‌شما به‌طور خودکار آکولادگذاری را انجام می‌دهد و از این خطای نابهنگام رهایی پیدا می‌کنید.

منابع

http://www.gimpel.com
http://valgrind.org
http://codeguide.googlepages.com/ codingtricks

شبکه‌های حسگر بی‌سیم

بقیه از صفحه ۱۳

یکی دیگر از مزایای این روش محافظت از مناطقی است که دارای شرایط جغرافیایی سخت هستند، به‌طوری که حضور انسان در آنجا مشکل یا غیرممکن باشد.

کشاورزی و دامداری

استفاده از شبکه‌های بی‌سیم حسگر در کشاورزی اجازه می‌دهد تا آبیاری به‌طور دقیق انجام شود و بارور کردن خاک با قرار دادن سنسورها در داخل خاک انجام می‌شود. برای این کار تعداد سنسور نسبتا کمی نیاز است (تقریبا یک سنسور در هر صد متر مربع) . به‌همین نحو، برای کنترل آفت در زمین کشاورزی می‌توان از این شبکه استفاده کرد.

کاربرد خانگی

هدف اصلی از شبکه‌های سنسوری در کاربردهای خانگی افزایش سطح راحتی و آرامش زندگی همراه با کاهش میزان مصرف انرژی است. از مهم‌ترین نمونه‌ها، کنترل دما و شرایط تهویه مطبوع، کنترل میزان مصرف برق، سیستم‌های ضد سرقت و در کل ایجاد خانه هوشمند، که موضوع مورد علاقه بسیاری از محققین و شرکت‌های تجاری است.

کنترل رفتار و حرکات کودک و جلوگیری از ورود کودک به قسمت‌های خطرناک منزل مثل آشپزخانه که امروزه با کمک فناوری بی‌سیم RFID پیاده‌سازی می‌شود.

دیگر کاربردها

برخی از کاربردهای شبکه‌های سنسوری، کاربردهای آزمایشگاهی، کاربردهای نظارتی، کاربردهای جهانگردی، کاربردهای آموزشی، کاربردهای حمل‌ونقل و کنترل ترافیک و و بسیاری موارد دیگر که هر روز بر تعداد آنها افزوده می‌شود.

1. Wireless Sensor Networks: WSN
and Atmospheric Administration: NOAA
2. National Oceanographic
3. Distributed Sensor Networks: DSN