bennyhuo · 更新于 2021-01-05

上一节 2-3 程序调试方法 【加餐】背景知识 下一节

代码风格介绍

虽然我们对 C 语言的认识还很少，不过我想要提前告诉大家的是，绝大多数语言的编写风格都是比较自由的，源代码中的空格、换行符、函数、变量的命名也都只有一个相对宽泛的约定，如果没有统一的标准，那么写出来的代码将难以阅读。

不信的话，请看这里：

代码块

```c
#include<stdio.h>
int a = 256;int main(){for(char b[a+a+a],
*c=b ,*d=b+ a ,*e=b+a+a,*f,*g=fgets(e,(b[
a]=b [a+a] =a- a,a) , stdin);c[0]=a-a,f=c
,c=d ,d=e ,e=f, f= g,g =0,g = fgets(e,a+a
-a+ a -a+a -a+ a- +a,stdin ),f +a-a ; pu\
tchar(+10)) { for( int h= 1,i=1,j, k=0 ,l
=e[0]==32,m,n=0,o=c [ 0]== 32, p, q=0;d[q
];j=k,k=l,m=n,n=o,p=(j)+(k* 2 )+(l =(i =
e[ q]&&i ) &&e[q +1 ]== 32,l*4)+(m* 8 )+(
16*  n  )+(  o  =(h =c[ q]&&h)&&c[q+1]==
32,o* (16+16) )+0-0 +0, putchar(" ......"
/*\  (  |||  )  |/|/ / */".')|)\\\\\\\\'"
"" "|||"   "|||" "|'" ")|)\\\\\\\\'/|/(/"
"(/'/|/\\|\\|'/|/(/(/'/|/\\|\\|"[d[q++]==
32?p:0]));}}/* typographic tributaries */
```

以上代码源自于国际 C 语言混乱大赛，虽然我们还没有开始学习 C 语言的语法，但本能地我们就能感觉出来上面的这代码是什么啊，乱七八糟的。不过，可别小瞧人家，人家可是能够正常编译运行的 C 语言代码。

好的代码风格除了让代码的书写方式更符合大家的习惯，还应该让代码更加能够表达其含义。对比下下面的代码：

代码块

```c
#include <stdio.h>

int main() {
  int a = 0;
  int i = 0;
  while(i < 10) {
    a = a + i;
    i++;
  }
  printf("%d\n", a);
  return 0;
}
```

这段代码其实目的是求 0 - 9 的整数的和，其中 a、i 都是后面我们即将要学到的变量的定义，while 则是循环语句，等大家学完这些语法之后就会发现这个程序是非常简单的，但我们仍然不能一眼就能搞清楚 a 这个变量的目的是什么。

代码块

```c
#include <stdio.h>

int main() {
  int sum = 0;
  int i = 0;
  while(i < 10) {
    sum = sum + i;
    i++;
  }
  printf("%d\n", sum);
  return 0;
}
```

我们只要稍微调整一下命名，将 a 改名为 sum，问题一下就清晰了，sum 我们都知道是和的意思，那么给变量命名的开发者似乎想要通过这个 sum 来告诉代码阅读者他在这里只是想求个和。

代码当中应当有适当的注释，注释在所有语言中都有，它们不是给机器看的，是代码开发者在代码之外添加的信息以供代码阅读者方便的了解代码的功能和意图。

C 语言在早期的 C90 当中，注释只支持 `/* ... */` 的方式，即：

代码块

```c
/* 下面的代码用于整数求和 */
int sum = 0;
int i = 0;
...
```

也可以多行：

代码块

```c
/* 下面的代码用于整数求和
   其中 sum 用于存储整数的和
   i 用于存储每一个整数的值
*/
```

当然，经常为了美观，我们还会在每行前面加一个 * 号：

代码块

```c
/* 下面的代码用于整数求和
 *  其中 sum 用于存储整数的和
 *  i 用于存储每一个整数的值
 * */
```

总而言之，遇到 `/*` 注释开始，遇到 `*/` 注释结束，不允许嵌套。

从 C99 开始，C 语言也支持了单行注释，这个也是 C++、Java 等常见的语言所支持的特性：

代码块

```c
// 打印输出整数之和
printf("%d\n", sum);
```

单行注释以 `//` 开头，整个这一行后面的内容都会作为注释。除了独占一行，单行注释也可以附加在代码之后：

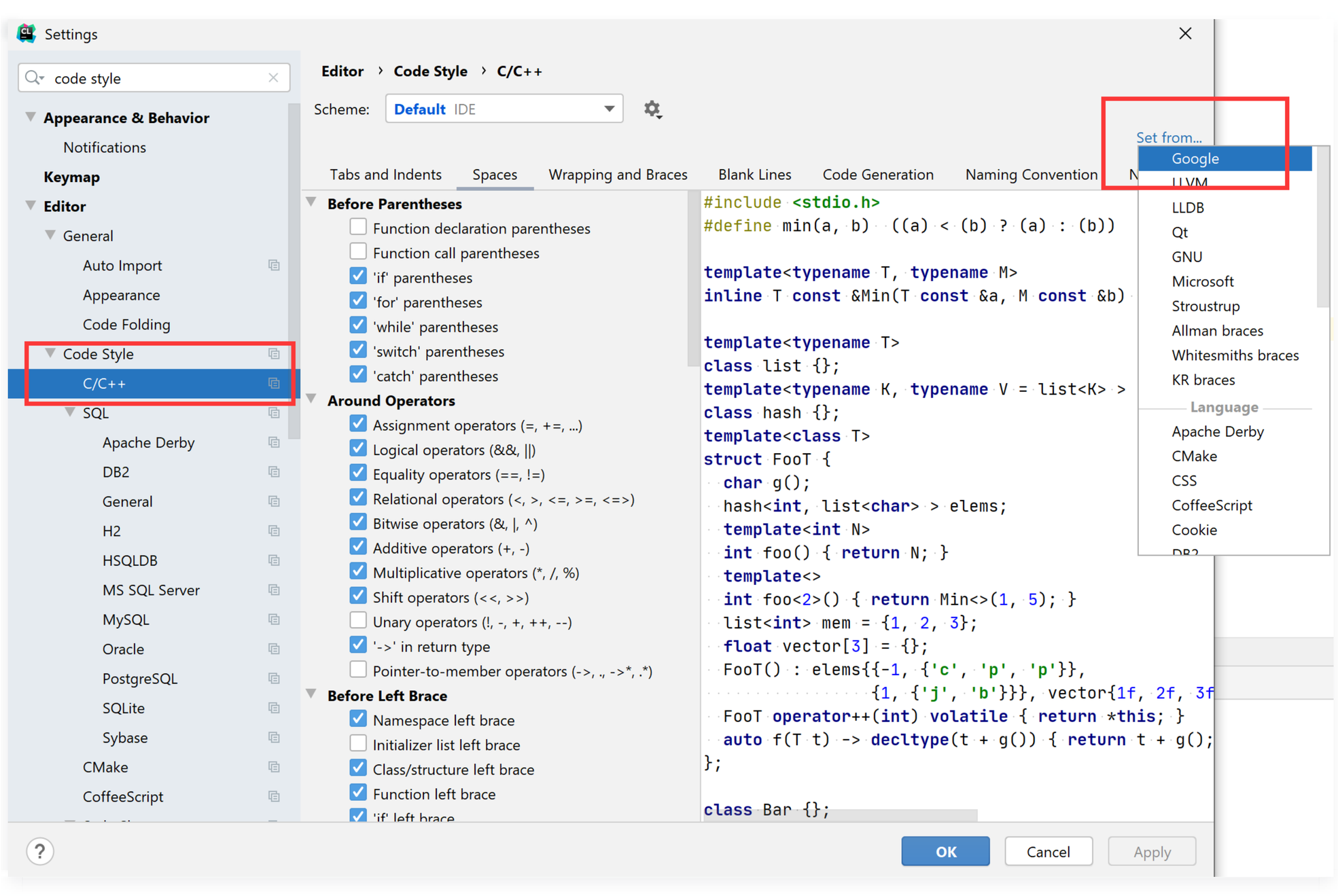
代码块

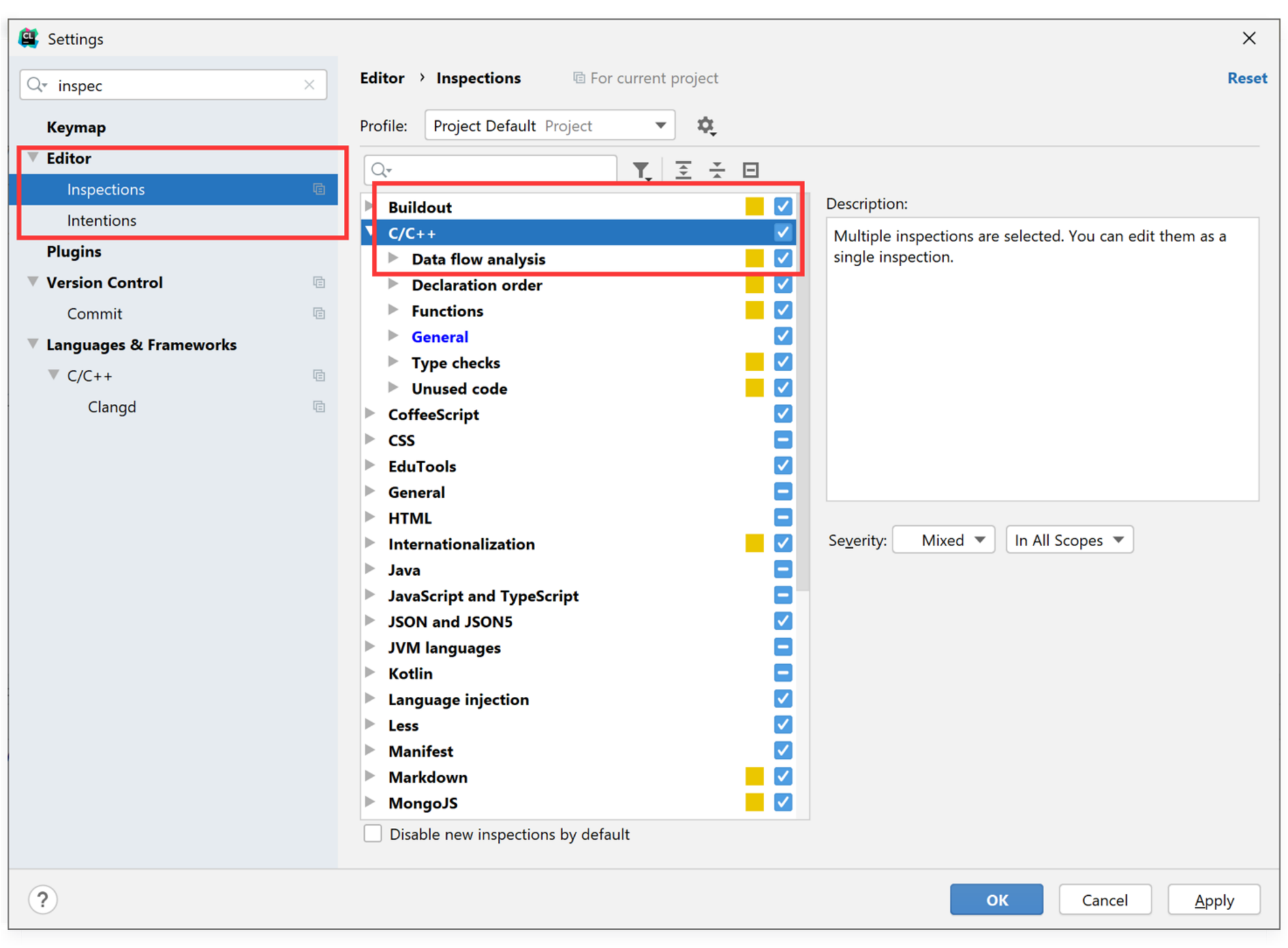
```c
printf("%d\n", sum); // 打印输出整数之和
```

注释在编译时会被替换为空格，因此可以出现在任意空格可以出现的位置。

我们从一开始就要培养编写好的代码的习惯和意识，实践证明，培养代码"洁癖"有助于激发程序员的上进心。

实际上，对于代码怎么写，早就有大公司给出了指导意见，我们在课程内容中使用 Google 的 C 语言代码规范来约束我们的课程示例代码。为了让聪明的 IDE 时刻对我们的不好的代码进行提示，我们在 CLion 当中直接应用 Google 的代码规范，操作方式如下图：





确保你的配置与截图一致之后，我们来看下效果：

我定义了一个函数 `printHello`，用的是驼峰命名法，首字母小写，后续单词首字母大写。这个命名法在 Java 当中常用，但不符合 C/C++ 开发者的习惯，Google 代码规范中要求使用帕斯卡命名法，即所有单词的首字母都要大写，此时 CLion 便提示我们应当将名字改成 `PrintHello`。

后面这些内容我们将在课程的讲授过程中不断看到，如果这时候大家不能完全理解，那是正常的，只需要有个大致的印象，并且知道自己一定要成为写高质量代码的开发者就对了。

2-3 程序调试方法 上一节　　下一节 【加餐】背景知识