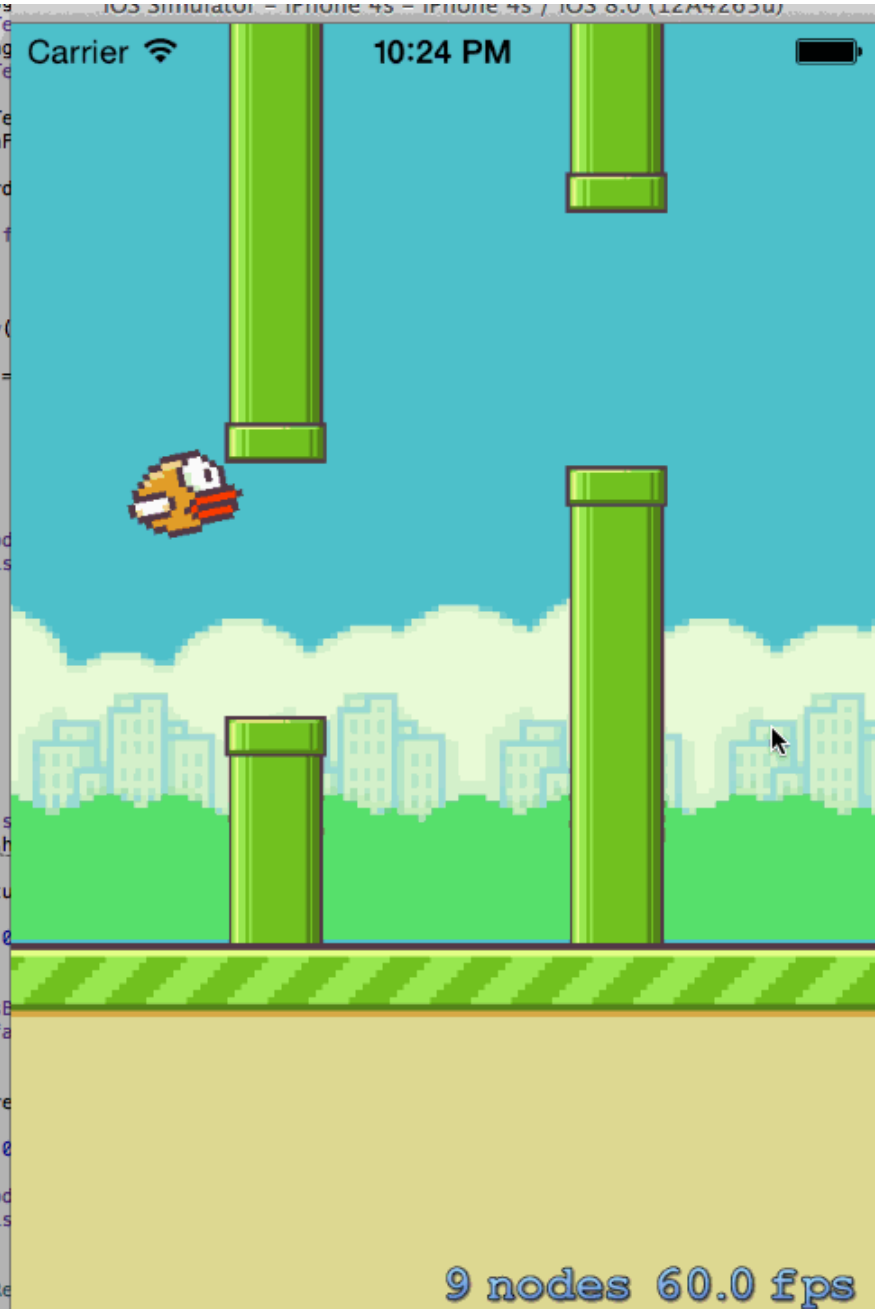
37 MB

Zero KB/s

Zero KB/s

```
        bird.physicsBody = SKPhysicsBody(
        bird.physicsBody.dynamic = true
        bird.physicsBody.allowsRotation =

        self.addChild(bird)

        // create the ground
        var ground = SKNode()
        ground.position = CGPointMake(0,
        ground.physicsBody = SKPhysicsBod
        ground.physicsBody.dynamic = fals
        self.addChild(ground)


    }

    func spawnPipes() {
        var pipePair = SKNode()
        pipePair.position = CGPointMake(
        pipePair.zPosition = -10;

        var height = UInt32( self.frame.s
        var y = arc4random() % height + h

        var pipeDown = SKSpriteNode(textu
        pipeDown.setScale(2.0)
        pipeDown.position = CGPointMake(0


        pipeDown.physicsBody = SKPhysicsB
        pipeDown.physicsBody.dynamic = fa
        pipePair.addChild(pipeDown)

        var pipeUp = SKSpriteNode(texture
        pipeUp.setScale(2.0)
        pipeUp.position = CGPointMake(0.0

        pipeUp.physicsBody = SKPhysicsBod
        pipeUp.physicsBody.dynamic = fals
        pipePair.addChild(pipeUp)

        pipePair.runAction(movePipesAndRe
```



Apple的Swift语言

发表于 2014-06-12 | 分类于 技术控 | 1 Comment | 阅读次数: 1445
字数统计: 981 字 | 阅读时长 ≈ 3 分钟

六月初的WWDC2014，可谓果粉们的盛会，除了新的操作系统带来的喜悦，我想，对于开发者们来说，更令人欣喜若狂的应该是苹果新推出的Swift语言了。

火爆的Swift

其实说新也不算新，只能说Apple的保密工作做得太好了。其实，早在2010年的时候，Apple已经让Chris Lattner(LLVM 项目的主要发起人与作者之一，Clang 编译器的作者)来打造这门新的语言了。直到在WWDC2014接近尾声的时候，大家都以为发布会即将结束的时候，Apple突然抛出了这么一个“重磅炸弹”，让所有的开发者们都为之一振。而Chris也在twitter开玩笑的说，他自己估计是世界上第一个拥有4年Swift编程经验的人……

Apple提到这门独特的Swift语言，是objective-c, without c，大有用Swift替代objective-c之意。这门新的Swift语言，迅速在各种社区得到热烈的讨论，其火爆程度实在是让人惊讶。同样的，也即在第二天，Github上就有了第一个用Swift移植的FlappyBird的项目，一瞬间，star数量猛涨…… 随即，各种编辑器的Swift插件就出来了，包括Sublime的，以及Vim的。后来比较有名的游戏2048也出了Swift的版本。

我们这门课希望大家锻炼

- **分析问题的思路**
- **设计解决方案的能力**
- **基于现有资源编写程序去解决问题的能力**
  - 提取报错信息中的关键内容，判断问题所在
  - 查阅资料，尝试解决bug
  - …
  - 本意不想和具体编程语言的特定语法绑定太密切

在线辅导和答疑

- 每周三晚上9-10点
- 腾讯会议：

孙泽群 邀请您参加腾讯会议
会议主题：2024秋季程序设计基础课程答疑辅导
会议时间：2024/10/16 21:00-22:00 (GMT+08:00)
中国标准时间 - 北京
重复周期：2024/10/16-2025/01/15 21:00-22:00，
每周 (周三)

点击链接入会，或添加至会议列表：
https://meeting.tencent.com/dm/I2nNFzkiIOE0

#腾讯会议：547-5302-8950

复制该信息，打开手机腾讯会议即可参与

C/C++的一些小知识

- **圆括号的作用**
  - 方法调用传参
  - 控制操作优先级?
  - 如何理解`while((ch=getchar()) != '\n')`中的三个圆括号呢?

- **赋值操作有返回值**
  - 赋值表达式的返回值为操作符右侧变量值

- **思考并尝试小例子**
  - **右侧代码的输出是什么?**
  - **两个赋值表达式的圆括号优先级谁高?**

  （我上学那时候，不同编译器会有不同结果）

  可以学完《编译原理》之后再回来看这个例子

```c
int main() {
    int b = 0;
    int a = (b = 1) + (b = 2);
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

C/C++的一些小知识

- **ASCII**
  - ASCII是基于拉丁字母的一套电脑字符编码标准。它主要用于显示现代英语，而其扩展版本延伸美国标准信息交换码则可以部分支持其他西欧语言，并等同于国际标准ISO/IEC 646。 美国信息交换标准代码是这套编码标准的传统命名，互联网号码分配局现在更倾向于使用它的新名字US-ASCII。

- **ASCII可显示字符**
  - 可显示字符编号范围是32-126，共95个字符。
  - 32～126(共95个)是字符(32是空格)，其中48～57为0到9十个阿拉伯数字。
  - 65～90为26个大写英文字母，97～122号为26个小写英文字母
  - 其余为一些标点符号、运算符号等。

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

C/C++的一些小知识

- **隐式类型转换/类型提升**
  - char -> int
  - 所以，整型变量和字符变量可以运算

```
char a;
scanf("%c", &a);
printf("%d\n", a);
printf("%c\n", a);
```

输出多少呢

  - float -> double

```
int main() {
    double a;
    scanf("%lf", &a);
    printf("%f\n", a);
    printf("%lf\n", a);
    return 0;
}
```

  - int -> double

https://www.runoob.com/cprogramming/c-data-types.html

C/C++的一些小知识

- **隐式类型转换/类型提升**
  - char -> int

```
char a;
scanf("%c", &a);
printf("%d\n", a);
```

输出多少呢

整数类型

下表列出了关于标准整数类型的存储大小和值范围的细节：

类型	存储大小	值范围
char	1 字节	-128 到 127 或 0 到 255
unsigned char	1 字节	0 到 255
signed char	1 字节	-128 到 127
int	2 或 4 字节	-32,768 到 32,767 或 -2,147,483,648 到 2,147,483,647
unsigned int	2 或 4 字节	0 到 65,535 或 0 到 4,294,967,295
short	2 字节	-32,768 到 32,767
unsigned short	2 字节	0 到 65,535
long	4 字节	-2,147,483,648 到 2,147,483,647
unsigned long	4 字节	0 到 4,294,967,295

https://www.runoob.com/cprogramming/c-data-types.html

C/C++的一些小知识

- **隐式类型转换/类型提升**
  - char – int

```
char a;
scanf("%c", &a);
printf("%d\n", a);
```

输出多少呢

- **思考右边的例子**
  - 打印的两行结果一样吗?
  - printf使用%lf正确与否?

```
int main() {
    double a;
    scanf("%lf", &a);
    printf("%f\n", a);
    printf("%lf\n", a);
    return 0;
}
```

https://cplusplus.com/reference/cstdio/printf/

C/C++的一些小知识

- **scanf函数**
  - 函数定义：`int scanf ( const char * format, ... );`
  - 格式符：

	specifiers					
length	d i	u o x	f e g a	c s [] [^]	p	n
(none)	int*	unsigned int*	float*	char*	void**	int*
hh	signed char*	unsigned char*				signed char*
h	short int*	unsigned short int*				short int*
l	long int*	unsigned long int*	double*	wchar_t*		long int*
ll	long long int*	unsigned long long int*				long long int*
j	intmax_t*	uintmax_t*				intmax_t*
z	size_t*	size_t*				size_t*
t	ptrdiff_t*	ptrdiff_t*				ptrdiff_t*
L			long double*			

  - 返回值：如果发生错误或者读到文件结尾，返回`EOF (-1)`

https://cplusplus.com/reference/cstdio/scanf/

C/C++的一些小知识

■ scanf可以读取空格吗

```c
int main() {
    char c;
    while (scanf("%c", &c) != EOF) {
        putchar(c);
    }
    return 0;
}
```

```c
int main() {
    char a[1000];
    scanf("%s", a);
    printf("%s", a);
    return 0;
}
```

输入带空格的字符串，按下回车，
看看左边两个代码的输出结果呢

C/C++的一些小知识

- printf函数
  - 格式符

	specifiers						
length	d i	u o x X	f F e E g G a A	c	s	p	n
(none)	int	unsigned int	double	int	char*	void*	int*
hh	signed char	unsigned char					signed char*
h	short int	unsigned short int					short int*
l	long int	unsigned long int		wint_t	wchar_t*		long int*
ll	long long int	unsigned long long int					long long int*
j	intmax_t	uintmax_t					intmax_t*
z	size_t	size_t					size_t*
t	ptrdiff_t	ptrdiff_t					ptrdiff_t*
L			long double				

https://cplusplus.com/reference/cstdio/printf/

C/C++的一些小知识

- **getchar函数**
  - 函数定义：`int getchar ( void );`
    - 可以读取空格' '和回车符'\n'
  - 返回值
    - 如果成功，返回标准输入流的下一个字符（自动提升类型为`int`）
    - 如果失败或者读到文件末尾，返回`EOF`
    - 用`shell`重定向进行文件输入试试?

https://cplusplus.com/reference/cstdio/getchar/

C/C++的一些小知识

- **cin函数**
  - 是一个对象，而不是方法
  - cin不读取空格字符，会自动跳过

```
int main() {
    char c;
    while (cin >> c) {
        if (c == ' ') {
            printf("读取到空格");
            break;
        }
        putchar(c);
    }
    return 0;
}
```

输入带空格的字符串，按下回车，看看上述代码的输出呢

std::cin

```
extern istream cin;
```

Standard input stream

Object of class istream that represents the ***standard input stream*** oriented to narrow characters (of type char). It corresponds to the ***C stream*** stdin.

The *standard input stream* is a source of characters determined by the environment. It is generally assumed to be input from an external source, such as the keyboard or a file.

As an object of class istream, characters can be retrieved either as formatted data using the extraction operator (operator>>) or as unformatted data, using member functions such as read.

https://cplusplus.com/reference/iostream/cin/

C/C++的一些小知识

■ 输入流操作符>>

Return Value

The `istream` object (`*this`).

The extracted value or sequence is not returned, but directly stored in the variable passed as argument.

Errors are signaled by modifying the ***internal state flags***, except for ***(3)***, that never sets any flags (but the particular manipulator applied may):

flag	error
`eofbit`	The input sequence has no more characters available (***end-of-file*** reached).
`failbit`	Either no characters were extracted, or the characters extracted could not be interpreted as a valid value of the appropriate type. For ***(2)***, it is set when no characters are inserted in the object pointed by *sb*, or when *sb* is a ***null pointer***.
`badbit`	Error on stream (such as when this function catches an exception thrown by an internal operation). When set, the integrity of the stream may have been affected.

Multiple flags may be set by a single operation.

https://cplusplus.com/reference/istream/istream/operator%3E%3E/

C/C++的一些小知识

- **键盘输入**
  - 结束符：回车，'\t'
- **文件输入**
  - 结束符：EOF
- **while(cin)?**

std::basic_ios<CharT,Traits>::operator bool

```
operator /* unspecified-boolean-type */() const;      (1)  (until C++11)
explicit operator bool() const;                        (2)  (since C++11)
```

Checks whether the stream has no errors.

1) Returns a value that evaluates to `false` in a boolean context if `fail()` returns `true`, otherwise returns a value that evaluates to `true` in a boolean context.

2) Returns `true` if the stream has no errors and is ready for I/O operations. Specifically, returns `!fail()`.

This operator makes it possible to use streams and functions that return references to streams as loop conditions, resulting in the idiomatic C++ input loops such as `while (stream >> value) {...}` or `while (std::getline(stream, string)) {...}`. Such loops execute the loop's body only if the input operation succeeded.

https://en.cppreference.com/w/cpp/io/basic_ios/operator_bool

CMS平台的OJ系统测评脚本

```bash
#!/bin/bash
cd WORKSPACE

challengeProgramNames=("p1.cpp" "p2.cpp" "p3.cpp" "p4.cpp")
input=$2; inputType=$3;OLD_IFS="$IFS"; IFS=,; ins=($input); type=($inputType); IFS="$OLD_IFS"

compile(){
        challengeProgramName=${challengeProgramNames[$1 - 1]}
        # 选择编译器
        if echo "$challengeProgramName" | grep -q -E '\.cpp$'
        then
            compileCommand="g++ -lm -lc -lz -Werror "
        else
            compileCommand="gcc -lm -lc -lz -Werror "
        fi

        # 清理上次评测的可执行文件
        rm -f ./executeFile.out &> /dev/null

        # 如果按关卡组织，编译此关目录下全部源文件
        if [ $challengeProgramName != ${challengeProgramName%%/*} ]; then
            challengeProgramName=${challengeProgramName%/*.c*}"/*.c*"
        fi

        # 获取编译结果
        compileResult=$($compileCommand $challengeProgramName -o executeFile.out 2>&1)
        compileCode=$?
        if [ $compileCode -eq 0 ]; then
            compileResult=$(echo -n "compile successfully" | base64)
```

CMS平台的OJ系统测评脚本

```
sed "s/\r//g" $inputFile | $executeCommand 1> "$2/$1/$index.out" 2> "$2/$1/error$index.out"
```

sed命令是一种文本处理工具，-i选项表示直接修改文件内容，而不是输出到终端或者新建一个文件。该命令中的s表示替换，/r表示回车符，/g表示全局替换。因此，sed -i 's/\r//g'的作用是将文件中所有的回车符替换为空，即删除所有回车符。

Linux tr 命令用于转换或删除文件中的字符。
tr指令从标准输入设备读取数据，经过字符串转译后，将结果输出到标准输出设备。

https://www.runoob.com/linux/linux-comm-sed.html

https://www.runoob.com/linux/linux-comm-tr.html

Shell脚本的一些小知识

- **输入输出重定向：>、<、>>、<<**

命令	说明
command > file	将输出重定向到 file。
command < file	将输入重定向到 file。
command >> file	将输出以追加的方式重定向到 file。
n > file	将文件描述符为 n 的文件重定向到 file。
n >> file	将文件描述符为 n 的文件以追加的方式重定向到 file。
n >& m	将输出文件 m 和 n 合并。
n <& m	将输入文件 m 和 n 合并。
<< tag	将开始标记 tag 和结束标记 tag 之间的内容作为输入。

https://www.runoob.com/linux/linux-shell-io-redirections.html

Shell脚本的一些小知识

- **管道**
  - 管道命令操作符是“|”，它处理经由前面一个指令传出的正确输出信息，也就是 standard output 的信息，然后传递给下一个命令，作为标准的输入 standard input。
  - 对于 standard error 信息没有直接处理能力。

- **C/C++源码编译与执行**
  - **编译C：** gcc example.c -o example.*out*
  - **编译C++：** g++ example.cpp -o example.*out*
  - **执行：** ./example.*out*

- **命令行文本编辑器：Vim**

除了大家更加细心之外，如何减少小错误呢?

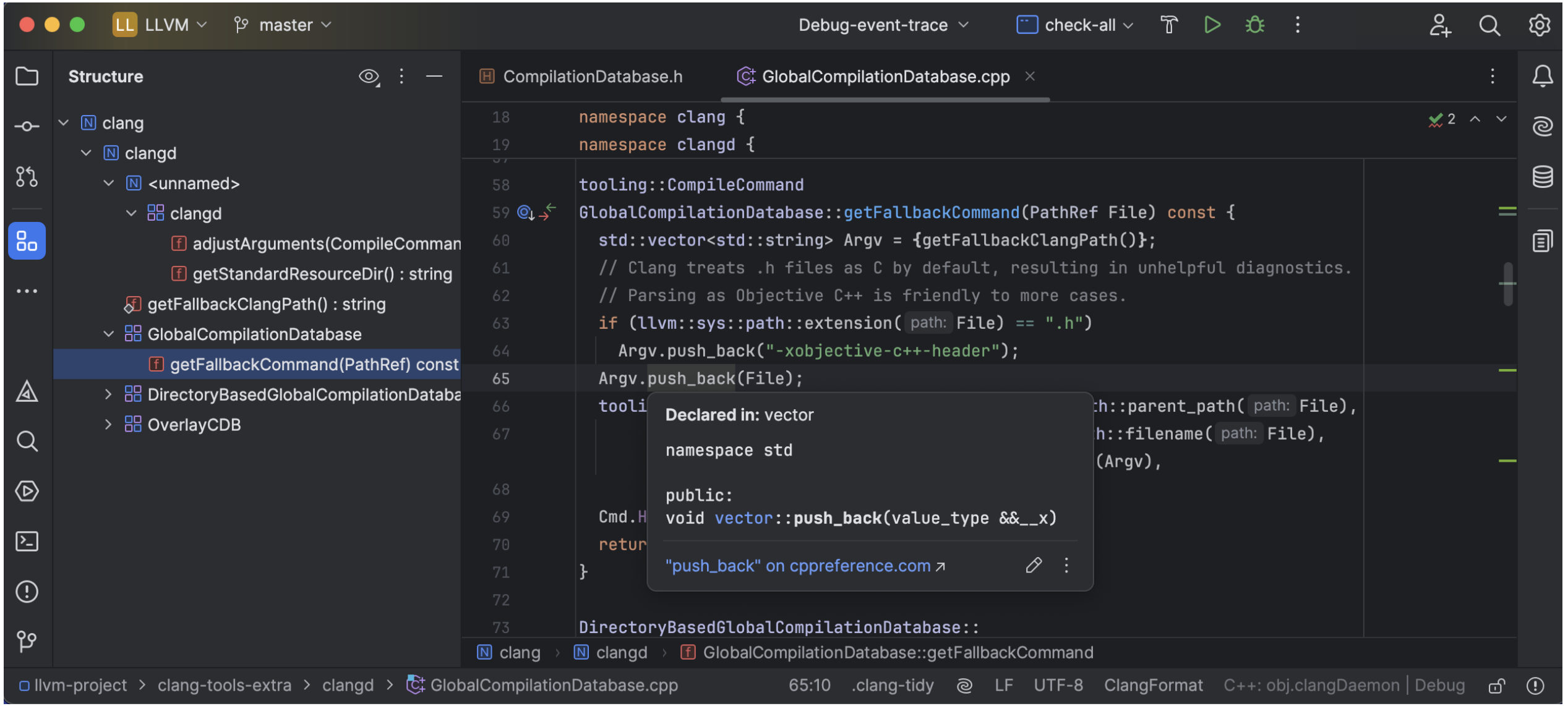
智能IDE推荐

- Windows平台：Visual Studio、**CLion**、VS Code配置插件
- MacOS平台：Xcode、 **CLion**、VS Code配置插件
- Linux平台： **CLion**、VS Code配置插件

- **CLion：** https://www.jetbrains.com/zh-cn/clion/
- 学生和老师可以凭借学校邮箱（带nju.edu.cn后缀）申请免费使用一年

- 更智能：
  - 整行代码补全、预测
  - 警告、错误提示更加细致
  - 更加智能的代码生成

智能IDE推荐


LLVM
master
Debug-event-trace
check-all
Structure
clang
clangd
<unnamed>
clangd
adjustArguments(CompileComman
getStandardResourceDir() : string
getFallbackClangPath() : string
GlobalCompilationDatabase
getFallbackCommand(PathRef) const
DirectoryBasedGlobalCompilationDataba
OverlayCDB
CompilationDatabase.h
GlobalCompilationDatabase.cpp
namespace clang {
namespace clangd {
tooling::CompileCommand
GlobalCompilationDatabase::getFallbackCommand(PathRef File) const {
std::vector<std::string> Argv = {getFallbackClangPath()};
// Clang treats .h files as C by default, resulting in unhelpful diagnostics.
// Parsing as Objective C++ is friendly to more cases.
if (llvm::sys::path::extension(path: File) == ".h")
Argv.push_back("-xobjective-c++-header");
Argv.push_back(File);
Declared in: vector
namespace std
public:
void vector::push_back(value_type &&__x)
"push_back" on cppreference.com
DirectoryBasedGlobalCompilationDatabase::
clang › clangd › GlobalCompilationDatabase::getFallbackCommand
llvm-project › clang-tools-extra › clangd › GlobalCompilationDatabase.cpp
65:10
.clang-tidy
LF
UTF-8
ClangFormat
C++: obj.clangDaemon | Debug

字符串循环输入演示

- OJ系统

```
char c;
while ((c = getchar()) != EOF) {
    //循环体
}
```

- 键盘输入

```
char c;
while ((c = getchar()) != '\n') {
    //循环体
}
```