Обзор Chaiscript и Учебник

Введение

Эта запись в блоге основана на Chaiscript 6.0.0. Этот отчет будет сосредоточен на самом хаи-скриптовом языке, API C++ не обсуждается.

Установка

Chaiscript требует среды компилятора g++/gcc. Готового пакета также нет, поэтому установка требует нескольких и разных шагов для Windows и Linux:

Установка Chaiscript для Windows

1. Установите Min-GW: Execute “mingw-get-setup.exe” от http://www.mingw.org/wiki/getting_started (http://www.mingw.org/wiki/getting_started).
2. Опционально установить ConEmu в качестве терминала для Min-GW: https://conemu.github.io/ (https://conemu.github.io/).
3. Установите cmake (Windows win64-x64 Installer (.msi)) от https://cmake.org/download/ (https://cmake.org/download/).
4. Скачать zip-файл Chaiscript из https://github.com/ChaiScript/ChaiScript/releases (https://github.com/ChaiScript/ChaiScript/releases). Сохранить zipfile в локальную папку (например. “C:\chai”) и извлеките zip-файл.

5. Откройте командное окно Min-GW (или ConEmu) и получите новую папку Chaiscript (которая содержит CMakeLists.txt).
   Примечание: “cd C:\chai” будет “кд/к/чай” в пределах Min-GW.
6. Выполнить cmake:
   cmake -G “MSYS Makefiles” -D MULTITHREAD_SUPPORT_ENABLED=FALSE CMakeLists.txt
7. Теперь создайте интерпретатор Chaiscript chai.exe выполнив сделать (в оболочке Min-GW).

Установка Chaiscript для Ubuntu Linux

1. Установите cmake (sudo apt install cmake).
2. Скачать zip-файл Chaiscript из https://github.com/ChaiScript/ChaiScript/releases (https://github.com/ChaiScript/ChaiScript/releases). Сохраните zipfile в локальную папку (, например. “ C: \ chai ”) и извлечь zip-файл.
3. Найдите папку Chaiscript (, которая содержит CMAkeLists.txt) и выполнить cmake:
   cmake -D MULTITHREAD_SUPPORT_ENABLED = ЛОЖНЫЙ CMAkeLists.txt
4. Выполнить сделать. chai.exe будет создан в той же папке.

Вызов Chaiscript

Вновь созданный исполняемый файл может быть протестирован по следующему вызову:
chai -c ‘dump_system()’

Это может зависеть от вашей системы, нужно ли вводить текст ./chai.exe или ./чай или просто чаи. Эта команда будет печатать все зарегистрированные функции Chaiscript.
Обычный пример HelloWorld будет выглядеть так:
chai -c ‘print(“Hello World!”);’

Конечно, команды Chaiscript могут быть помещены в файл:

my_first_script.chai

```
печать("Привет миру!");
```

Затем выполнить:
chai my_first_script.chai

Резюме:

- Одна строка кода Chaiscript может быть выполнена непосредственно с “-c” вариант.
- Файлы Chaiscript выполняются по вызову чаи с именем файла в качестве аргумента.

Chaiscript “print” Command

Все, что можно преобразовать в строку, может быть напечатано с помощью “печатать” функций. Вот еще примеры:

```
var v = 3;
печать("v=" + to_string(v));
печать("v=${v}");
```

- Для тестирования сохраните код в файл и выполните файл (смотрите выше). Вы также можете протестировать одну строку кода, как это:
  chai -c ‘var v = 3; print(“v=” + to_string(v));’
- “var v = 3;” объявляет переменную v и присваивает ей значение 3.
- Оператор “+” соединяет две струны в примере выше.
- Выражение ${xyz} оценивает ксиз и вставляет результат оценки в результат.
- печать() всегда печатает обратную/строчную подачу каретки.
- Заявления должны быть прекращены с “;“. Я много случаев Chaiscript терпим, если “;” там нет.

Резюме

- Используйте печать() функция для печати строк и значений.
- Используйте to_string() преобразовать что-то в строку
- Результаты выражения могут быть встроены в строку с ${xyz}

Условия и Петли

“if” Заявление

```
var v = 3;
если (v != 3 )
{
    печать("v=" + to_string(v));
}
другой
{
    принт("три");
}
```

- “if” утверждения похожи на C/C++ или JavaScript.
- В Chaiscript всегда ставится if-блок и/или else-блок между “{” и “}“. Также одно утверждение должно быть помещено между “{” и “}“.

- Вот краткое изложение операторов сравнения:

Оператор	Описание
==	Тест для “equal”
!=	Тест для “not equal”
<=	Нижняя или равная
>=	Большие или равные
<	Нижний
>	Большой

“for” Loop

```
var i;
для(i = 0; i < 5; ++i )
{
    печать(i);
}
```

- “for” loop похож на C/C++ “for” loop: Он содержит три части, утверждение для инитализации петлевой переменной, условие испытания для петлевой переменной и действие для модификации петлевой переменной.
- Приращение “++” и дряхление “—Оператор ” должен быть размещен перед переменной.

“while” Loop

```
var i;
i = 0;
пока(i < 5 )
{
  печать(i);
  ++i;
}
```

Петли можно остановить с помощью разрыв заявление:

```
var i;
i = 0;
пока(правда)
{
  ++i;
  если ( i > 5 )
  {
    перерыв;
  }
}
```

Резюме

- Chaiscript поддерживает если/другой, за и пока петли.

Объекты Build-In

Chaiscript включает в себя полезные объекты. Эти объекты перечислены здесь:

https://codedocs.xyz/ChaiScript/ChaiScript/namespaceChaiScript__Language.html (https://codedocs.xyz/ChaiScript/ChaiScript/namespaceChaiScript__Language.html)

В этой главе будет обсуждаться “карта“, “вектор” и “string” объекты.

“map” Объект

```
var m = ["a":111, "c":333];
печать(м);
```

- Объект карты может содержать набор пар ключ/значение.
- Ключ должен быть строкой.
- Ключ должен быть уникальным.
- Значение может быть любого типа.

Существует специальный “for”-петля для петли над всеми элементами карты:

```
var m = ["a":111, "b":222];
for( i : m ) // "i" будет петлять по всем парам ключ/значение
{
    print(i.second()); //печать значения пары ключ/значение
}
```

- Объявление петлевой переменной “i” не требуется.
- Использование пара.первый() для доступа к ключу пары ключ/значение
- Использование парасекунда() для доступа к значению пары ключ/значение.

В следующей таблице приведены некоторые примеры того, что вы можете сделать с картами:

Синтаксис	Описание	Пример
m = Map()	Создать новую пустую карту	
м[ключ]	Верните значение для ключа	var m = [“a”:111, “c”:333]; печать(m[“c”];
m[key] значение =	Вставить или перезаписать пару ключ/значение.	var m = Map(); m[“a”] = 111;
m[key].is_var_null()	Проверьте, недействителен ли ключ. Возвращает истину или ложный. Не используйте это: Лучше использовать функцию.count().	если (m[“a”].is_var_null() == true)
m.count(ключ)	Проверьте, как часто ключ присутствует на карте (Должно быть 0 или 1).	если (m.count(“a”) > 0)
м.size()	Возвратное число пар ключ/ значение.	var m = [“a”:111, “c”:333]; печать(m.size();
м1.вставка(м2)	Присоединяйтесь к двум картам m1 и m2 и поместите результат в m1.	var m = [“a”:111, “c”:333]; m.insert([“b”:222,”d”:444]; печать(м);
m.erase(клавиша)	Удалить пару ключ/значение с указанным ключом.	
м.clear()	Удалить все элементы с карты.	м.clear(); печать(m.size();
м.пустой()	Вернуть true, если карта пуста.	
м.range()	Возвращает объект дальности для петлирования по элементам.	
м.range().front()	Получить первую пару ключ/ значение.	
m.range().front().first()	Получить ключ первой пары ключ/значение (строка).	
m.range().front().second()	Получить значение первой пары ключ/значение.	

“vector” Объект

Вектор - это упорядоченный список объектов. В качестве элементов вектора могут использоваться любые объекты.

```
var v = ["a", 1, 2,9];
печать(v);
```

Доступ и изменение функций для вектора:

Синтаксис	Описание	Пример
v = Vector()	Создать новый пустой вектор	
в[п]	Возвращают объект в положение р. в[0] является первым элементом.	var v = [“a”, “c”]; печать(v[1];
v[p] значение =	Заменить объект на положение р. Примечание: Можно заменить только объекты одного типа.	var v = [“a”, “c”]; v[1] = “b”;
v.insert_at(p, значение)	Вставить значение перед объектом в положении р.	var v = [7, 8]; v.insert_at(1, 123);
v.erase_at(p)	Удалить объект в положении р. Общее количество объектов в векторе уменьшается на единицу.	var v = [“a”, “c”]; v.erase_at(0);
v.size()	Верните количество элементов в векторе.	var v = [“a”, “b”]; печать(v.size();
v.clear()	Удалить все элементы из вектора.	м.clear(); печать(m.size();
v.empty()	Вернуть true, если вектор пуст.	
м.range()	Возвращает объект дальности для петлирования по элементам.	
v.pop_back()	Удалить последний объект в векторе. То же, что и “v.erase_at(v.size()-1)“	
v.push_back(значения)	Добавить значение в конце вектора. То же, что и “v.insert_at(v.size(), значение)“	

“string” Объект

Строка - упорядоченный список чар. Функции доступа очень похожи на ‘Vector’ объект.

Строка определяется и назначается

```
var s = "abc";
```

Строка может пересекать несколько строк:

```
var s ="abc
деф";
```

Многие строковые операции принимают или возвращают значение типа ‘char’. Используйте следующие функции для преобразования из и в тип ‘char’:

- ‘A’: Объект типа ‘char’ с прописным A.
- чар (65): Объект типа ‘char’ с прописным A.
- int(‘A’): Целое значение 65. Примечание: to_int() не работает.
- to_string(‘A’): Строка с одной буквой “A”. Примечание: string(‘A’) не работает.
- to_char(” ABC”): Возвращает первый none whitespace char строки, ‘A’ в этом случае.

Синтаксис	Описание	Пример
s = string()	Создать новую строку.	
var s = “”	Создать новую строку.	var s = “a”; печать(ы);
s.size()	возвращает количество чар в строке.	var s = “abc”; печать(s.size();
c[п]	Доступ к шар в указанном положении ‘p’. ‘p’ начинается с 0. Возвращает ‘char’.	var s = “abc”; печать(s[0];
s.erase_at(p)	Удалите шар в указанном положении ‘p’.	var s = “abc”; s.erase_at(0);
s.insert_at(p, c)	Вставить шар ‘c’ в указанном положении ‘p’.	var s = “abc”; s.insert_at(1, ‘A’);
s.substr(p, cnt)	Возвращает ‘cnt’ чарс как струна, начиная с позиции ‘p’. ‘p’ должен быть ниже или равен ‘s.size()’. ‘p+cnt’ может быть больше, чем ‘s.size()’.	var s = “abc”; печать(s.substr(0), 1, 2);
s.clear()	Опустите струну.	s.clear(); печать(s.size();
с.пустой()	Вернуть true, если вектор пуст.	
s + t	Конкатенат двух струн.	
s += t	Прибавить строку ‘t’ к строке ‘s’.	
s.push_back(c)	Прибавить шар ‘c’ в конце строки.	var s = “abc”; s.push_back(‘\n’);
найти(с, п) s.find(p)	Найдите строку ‘p’ внутренняя струна ‘s’. Возвращает положение ‘p’ внутри ‘s’. Возврат 4294967295 если ‘p’ не может быть найден в ‘s’.	find(“abc”, “b”); // 1
to_int(s)	Интерпретирует строку ‘s’ как число и возвращает значение как целое число.	печать(to_int(” 1234″));

Вот полная ссылка на преобразование char, int и string. В таблице показан результат функции в самом левом столбце, применённый к значению в первой строке. Некоторые комбинации между значением и функцией могут привести к ошибке (в 6.0.0).

Обновление: Я добавил результаты из Chaiscript 6.1.0 после слэша, если результат отличается от предыдущей версии.

Функция	‘A’	65	“A”
шар	‘A’	‘A’	ошибка
to_char	ошибка / ‘A’	ошибка / ‘A’	‘A’
инт	65	65	ошибка
to_int	ошибка / 65	ошибка / 65	0
строка	ошибка	ошибка	“A”
to_string	“A”	“65”	“A”

Резюме

- Chaiscript поддерживает карты, векторы и строки
- Используйте цикл “for(:)” для доступа ко всем элементам карты, вектора или строки

Функции

Пример:

```
def my_функция (x)
{
    вар y;
    y = 3;
    возврат x*y;
}
```

- Определения функций начинаются с ключевого слова „def“.
- Значения возвращаются через ключевое слово „return“.
- Ключевое слово „return“ также останавливает выполнение функции и перескакивает обратно к вызывающей функции.
- Аргументы могут быть предварены именем типа: def my_function (в x)

Полезными типами являются:

Тип	Строительство	Функция
инт	var x = int(123); var x = 123;	def fn(int x)
int64_t	var x = int64_t(123);	def fn(int64_t x)
двойной	var x = двойной(1.23); var x = 1.23;	def fn(двойной x)
строка	var x = string(“abc”); var x = “”;	def fn(строка x)
Карта	var x = Map(); var x = [“a”:111, “b”:222];	def fn(Карта x)

Тип	Строительство	Функция
Вектор	var x = Vector(); var x = [];	def fn(Вектор x)

Резюме

- Используйте “дефлект” ключевое слово для определения функции.
- В отличие от нормальных переменных, аргументу функции может предшествовать имя типа.

Класс

```
класс MyClass
{
    var x; // переменная члена
    def MyClass() // конструктор (то же имя, что и класс)
    {
        this.x = 5; // init ваши переменные члена здесь
    }
    def show()//функция участника
    {
        печать("x=" + this.x.to_string();
    }
}

// тест
var o = MyClass();
o.show(); //это будет печатать x=5
```

- Класс определяется с помощью “класс” ключевое слово, за которым следует название класса.
- Функции члена и переменные члена определяются внутри класса.
- Специальная функция-член, имеющая то же имя, что и класс, всегда вызывается перед тем, как какая-либо другая функция будет вызвана (конструктором).
- Класс MUST имеет функцию конструктора.
- Внутри функций члена префиксные переменные члена с “это.“.