Встроенные типы, инструкции и операторы

Built-in types, casts, operators, C-strings, structs

Задачи занятия

- Изучить си строки.
- Понять различия между разными видами cast’ов и правильно выбирать нужный.
- Разработать программу с “искусственным интеллектом”.

Си строки

char str[] = “foo bar buzz”;

//’f’, ‘o’, ‘o’, ’ ‘, ‘b’, ‘a’, ‘r’, ‘ ‘, ‘b’, ‘u’, ‘z’, ‘z’, ‘\0’

Си строки

#include <string.h>

- size_t strlen(const char *s);
- char *strcpy(char *dest, const char *src);
- char *strncpy(char *dest, const char *src, size_t n);
- void *memcpy(void *dest, const void *src, size_t n);

Приведение типов: static_cast

static_cast<T2>(T1)

- Только безопасное (почти) приведение значения типа T1 в T2
- Иначе - ошибка компиляции
- T1 должен уметь как-то конвертироваться в T2 (определено встроенное или пользовательское преобразование)
- Также используется для конвертации из/в void*

Приведение типов: reinterpret_cast

reinterpret_cast<T2*>(T1*)

- Приведение указателя любого типа T1 к указателю любого типа T2
- Приведение интегральных типов к указателям и обратно
- Значение указателя/int не изменяется
- Не происходит изменения (конвертации) данных, на которые указывает указатель
- Не генерирует процессорных инструкций, является директивой для компилятора

static_cast, reinterpret_cast

```
float f = 1.25;
std::cout << (int)f << " " << static_cast<int>(f)
<< " " << *reinterpret_cast<int*>(&f) << std::
endl;
```

1 1 1067450368

Приведение типов: const_cast (for pointers, refs)

```
const char *const_str = “Copyright © 2014 blah blah corp”;
char *mutable_str = const_cast<char*> (const_str);

//segfault as const_str is in .rodata
mutable_str[2] = 3; //undefined behavior
```

C-style cast

- Пробует const_cast
- Пробует static_cast
- Пробует reinterpret_cast
- Ничего не подошло - ошибка

Пролог задач.

Разрабатываем бота-помощника, который умеет исполнять ваши команды. Ваша программа при запуске выводит приветствие и ожидает ввода от пользователя. Есть набор постоянно расширяющихся команд, эти команды имеют различный формат, принимают различное число аргументов, но начинаются всегда с одной буквы - идентификатора команды. Затем следуют аргументы через пробел или перенос строки

Что точно нужно использовать?

- enum
- c strings
- switch
- циклы
- функции
- потоки ввода/вывода cin/cout

Задача 1.

Наш бот делает первые шаги, а именно учится складывать и вычитать целые числа. Он хранит в памяти конечный результат, равный нулю по умолчанию. Вы можете приказать добавить число: “a 10” или вычесть число “s 11”, после этого он покажет вам результат всех операций по команде “r”.

a 10

s 11

r

<< -1

a 1

Задача 2.

Целые числа это просто, добавьте прибавление чисел с плавающей точкой двойной точности к тому же целочисленному результату, отбрасывая нецелую часть. Используйте static_cast для получения целочисленного значения из double.

a 5

f 2.3

r

>>7

Задача 3.

Отличное начало, пора научить бота работать с близкими к человеку сущностями, например словами.

Пусть вы вводите слово, а бот вам ссылку на результаты поиска в гугле. Ну или несколько слов (читайте до конца строки). Не забудьте заменить пробелы!

g How to create buffer cpp

>> https://www.google.ru/search?q=How%20to%20create%20buffer%20cpp

Задача 4.

Боту стало скучно и он хочет попрактиковаться в алгоритмах. Научите его искать самый большой палиндром в строке:

p cababad

>> ababa

Задача 5.

Самое время взламывать сервер института и увеличивать сумму своей стипендии, а мы не готовы. Научите бота печатать на экран кусок памяти, переданный по указателю.

m 0x20142015 1024

>> [1024 символа char]

Задача 6.

Научите бота печатать только валидные адреса в памяти.

v 0x20142015 1024

>> {Адрес начала валидного диапазона 1} {длина 1}: символы
>> {Адрес начала валидного диапазона N} {длина N}: символы