Algorithm

constructor:

String (const char *)

member:

String { ..... private:
char * elements;
char * end;
std::allocator <char> alloc;
}
;

1st step

char [ ] it ——— const_cast<char *> (it) ———> char * it

2nd step

find it1's begin and end auto it2 = it1

while (it2) it2++ ⇒ b it1 (e = it2++) [1]

3rd step take b, e as parameters. in range_initializers

prepare to return → pair (char *, char *) (name it str.)

(sub1: alloc_n_copy (b, e))

sub1_1: allocate (e-b) return str.first

sub1_2: uninitialized_copy (b, e, str.first) → str.second.

sub 2: takes the str.

elements = str.first end = str.second.

(sub1 & 2: range_initializer)

[1] Notice that a C-style string needs a '\0' at the end of its content.

~String();

{ free() }

void free() if(elements)

for_each(elements, end, [this](char&a){alloc.destroy(&a);})

alloc.deallocate(elements, end-elements);

obj for char* pointers,

uninitialized

```
void f(int *a){ *a = 35; }   <- to define
int c
    int *b = &c
    f(&b)   <- to call
```

pass by a pointer.