Don’t sacrifice the API to speed

C++ Summit 2020, China

dr Ivan Čukić

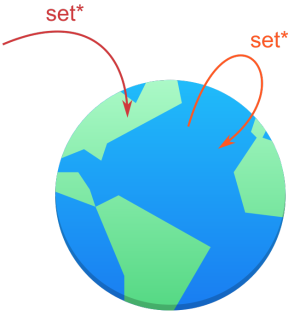
KDAB
ivan.cukic@kdab.com, ivan@cukic.co
https://kdab.com, https://cukic.co

About me

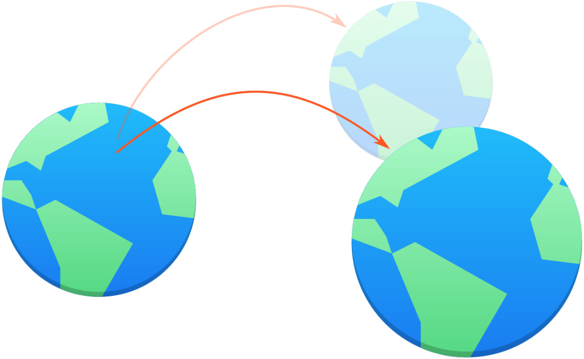
- KDAB senior software engineer
  Software Experts in Qt, C++ and 3D / OpenGL
- Author of the ”Functional Programming in C++” book
  available in English, Chinese, Korean, Russian, Polish
- Trainer / consultant
- KDE developer
- University lecturer

FAR AWAY

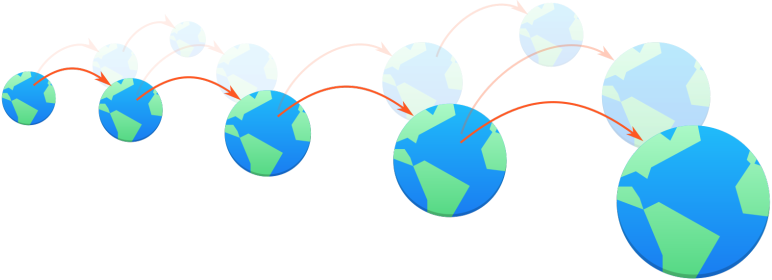
Far away worlds



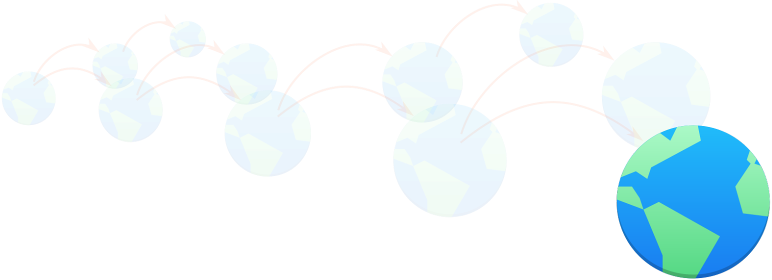
Far away worlds



Far away worlds



Far away worlds



Far away worlds

> Values belonging to a linear type must be **used exactly once**: like the world, they can not be duplicated or destroyed. Such values require no reference counting or garbage collection...
>
> Linear types can change the world!
> Philip Wadler

ATTACK OF THE CLONES

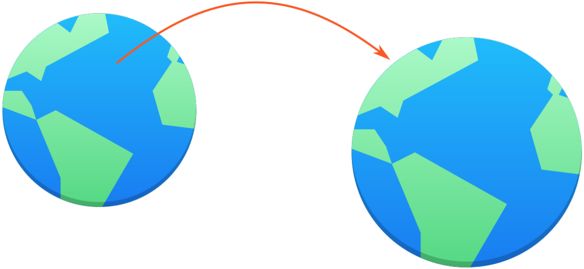
RAII

```
}
```

3
4
5

Clones

Clones



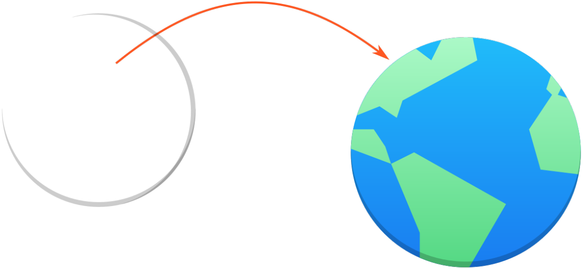
Clones

Clones

&&

Clones

Clones



Moves

- Resource ownership transfer
- Optimization – copy vs move
- API documentation / usage restriction

Moves

- Resource ownership transfer
- Optimization – copy vs move
- API documentation / usage restriction

Moves

```
void foo(type&& v)
{
    ...
}
```

Moves

```
class type {
    void foo( ) &&
    {
        ...
    }
}
```

*this is a temporary

Moves

```
type&& foo( )
{
    ...
}
```

Moves

```
type&& foo(type&& v)
{
    ...
}
```

Concepts and constraints

How to enforce moves with function templates?

```
template <typename T>
void foo(T&& val)
{
    ...
}
```

RUN DMC

Concepts and constraints

```
template <typename T>
constexpr bool is_int_v = std::is_same_v<T, int>;
```

Concepts and constraints

```
template <typename T>
concept IsInt = std::is_same_v<T, int>;
```

> not a proper concept,
> demonstration purposes only!

Concepts and constraints

```
template <typename T>
    requires (IsInt<T>)
void foo(T&& v)
{
    ...
}
```

Concepts and constraints

```
template <typename T>
    requires (is_int_v<T>)
void foo(T&& v)
{
    ...
}
```

Clones

```
template <typename T>
    requires (???)
void foo(T&& v)
{
    …
}
```

Clones

```
typedef T&  lref;
typedef T&& rref;

T value;

lref&  r1 = value; // type of r1 is T&
lref&& r2 = value; // type of r2 is T&
rref&  r3 = value; // type of r3 is T&
rref&& r4 = T();   // type of r4 is T&&
```

Clones

```
template <typename T>
    requires (!std::is_lvalue_reference_v<T>)
void foo(T&& v)
{
    ...
}
```

API

API

```
std::getline(std::cin, s);
```

API

```
std::string&& getline(std::istream& in,
                      std::string&& buf);

s = getline(std::cin, std::move(s));
```

Attack of the clones

```
istream_sequence<std::string> in{std::cin};

std::string result;
for (const auto& token: in) {
    result.append(token);
}
```

Attack of the clones

```
istream_sequence<std::string> in{std::cin};

std::string result;
for (const auto& token: in) {
    result.append(token);
}
```

Remember what Sean said?

Attack of the clones

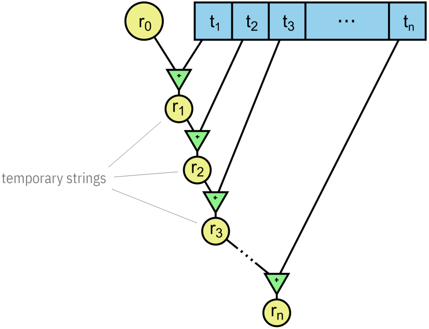
```
istream_sequence<std::string> in{std::cin};

const auto result =
          accumulate(in, string{});
```

Remember what Sean said?

Attack of the clones

```
template <typename InputIt, typename T>
T accumulate(InputIt first, InputIt last, T init)
{
    while (first != last) {
        init = init + *first;
        ++first;
    }
    return init;
}
```

Attack of the clones



Attack of the clones

```
template <typename InputIt, typename T>
T accumulate(InputIt first, InputIt last, T init)
{
    while (first != last) {
        init = std::move(init) + *first;
        ++first;
    }
    return init;
}
```

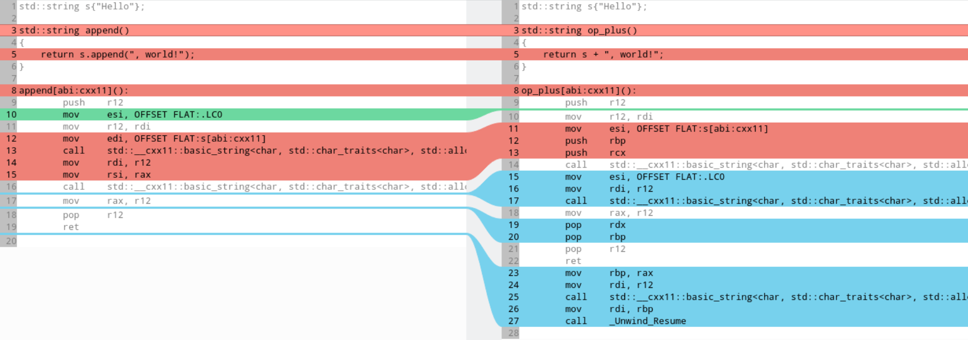
Attack of the clones

Copying is the silent (performance) killer

PERFORMANCE

Assembly

```
std::string s{”Hello”};

std::string append()
{
    return s.append(”, world!”);
}

std::string op_plus()
{
    return s + ”, world!”;
}
```

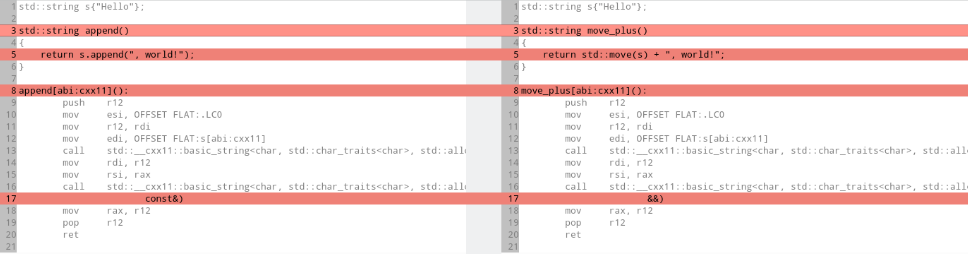


```
std::string s{"Hello"};

std::string append()
{
    return s.append(", world!");
}

append[abi:cxx11]():
        push    r12
        mov     esi, OFFSET FLAT:.LC0
        mov     r12, rdi
        mov     edi, OFFSET FLAT:s[abi:cxx11]
        call    std::__cxx11::basic_string<char, std::char_traits<char>, std::all
        mov     rdi, r12
        mov     rsi, rax
        call    std::__cxx11::basic_string<char, std::char_traits<char>, std::all
        mov     rax, r12
        pop     r12
        ret
```

```
std::string s{"Hello"};

std::string op_plus()
{
    return s + ", world!";
}

op_plus[abi:cxx11]():
        push    r12
        mov     r12, rdi
        mov     esi, OFFSET FLAT:s[abi:cxx11]
        push    rbp
        push    rcx
        call    std::__cxx11::basic_string<char, std::char_traits<char>, std::all
        mov     esi, OFFSET FLAT:.LC0
        mov     rdi, r12
        call    std::__cxx11::basic_string<char, std::char_traits<char>, std::all
        mov     rax, r12
        pop     rdx
        pop     rbp
        pop     r12
        ret
        mov     rbp, rax
        mov     rdi, r12
        call    std::__cxx11::basic_string<char, std::char_traits<char>, std::all
        mov     rdi, rbp
        call    _Unwind_Resume
```

Assembly

```
std::string s{"Hello"};

std::string append()
{
    return s.append(", world!");
}

std::string move_plus()
{
    return std::move(s) + ", world!";
}
}
```



```
std::string s{"Hello"};

std::string append()
{
    return s.append(", world!");
}

append[abi:cxx11]():
        push    r12
        mov     esi, OFFSET FLAT:.LC0
        mov     r12, rdi
        mov     edi, OFFSET FLAT:s[abi:cxx11]
        call    std::__cxx11::basic_string<char, std::char_traits<char>, std::all
        mov     rdi, r12
        mov     rsi, rax
        call    std::__cxx11::basic_string<char, std::char_traits<char>, std::all
                       const&)
        mov     rax, r12
        pop     r12
        ret
```

```
std::string s{"Hello"};

std::string move_plus()
{
    return std::move(s) + ", world!";
}

move_plus[abi:cxx11]():
        push    r12
        mov     esi, OFFSET FLAT:.LC0
        mov     r12, rdi
        mov     edi, OFFSET FLAT:s[abi:cxx11]
        call    std::__cxx11::basic_string<char, std::char_traits<char>, std::all
        mov     rdi, r12
        mov     rsi, rax
        call    std::__cxx11::basic_string<char, std::char_traits<char>, std::all
                       &&)
        mov     rax, r12
        pop     r12
        ret
```

Value Proposition:

Allocator-Aware (AA) Software

John Lakos

Saturday, April 13, 2019

This version is for ACCU'19.

Returning values

- (N)RVO – result is constructed in the caller
- Moved to the caller (CWG 1579)
- Copied into the caller

CWG 1579

Currently the conditions for moving from an object returned from a function are tied closely to the criteria for copy elision, which requires that the type of the object being returned be the same as the return type of the function. Another possibility that should be considered is to allow something like

```
optional<T> foo() {
    T t;
    ...
    return t;
}
```

and allow `optional<T>::optional(T&&)` to be used for the initialization of the return type. **Currently this can be achieved explicitly by use of std::move, but it would be nice not to have to remember to do so**.

Returning values

```
U fun()
{
    T value;
    …
    return value; // move constructed
}
```

Assembly

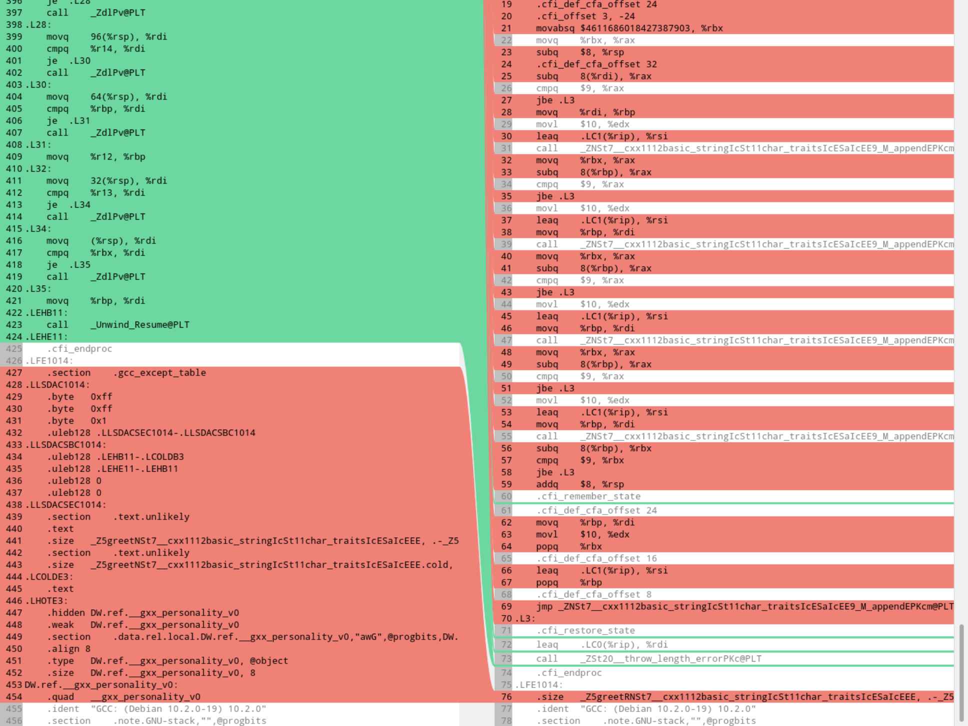
Better than RVO?

/tongue-in-cheek/

Returning values

```
std::string hello(std::string val)
{
    val.append(”Hello C++!”);
    return val;
}

std::string greet(std::string s)
{
    return hello(hello(hello(hello(hello(s)))));
}
```

```
void hello(std::string& val)
{
    val.append(”Hello C++!”);
}

void greet(std::string& s)
{
    hello(s);
    hello(s);
    hello(s);
    hello(s);
    hello(s);
}
```



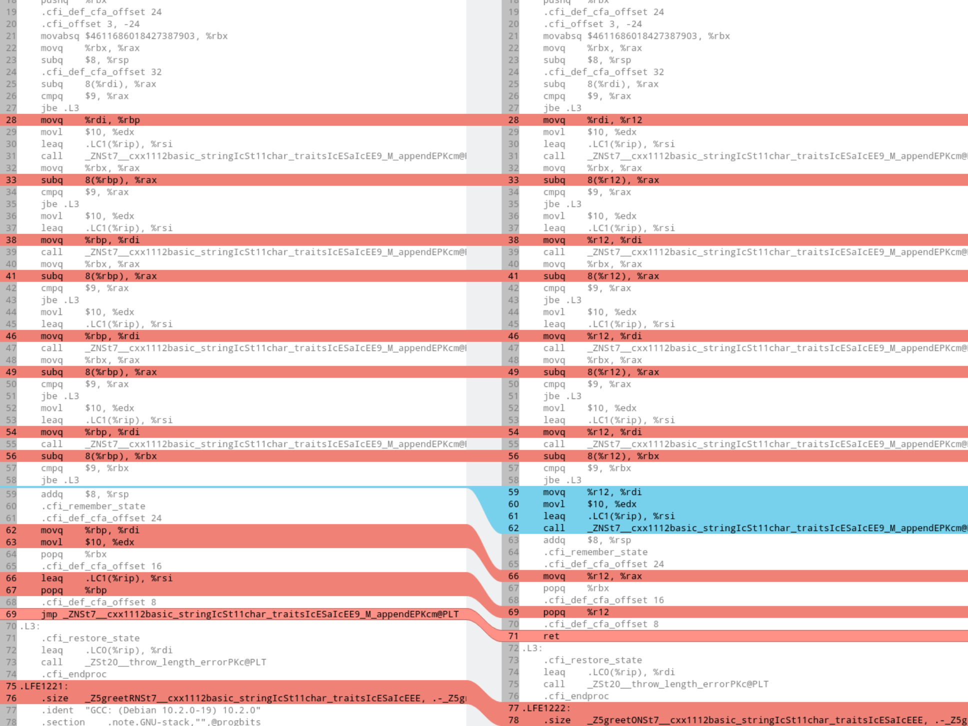
```
	je	.L28
	call	_ZdlPv@PLT
.L28:
	movq	96(%rsp), %rdi
	cmpq	%r14, %rdi
	je	.L30
	call	_ZdlPv@PLT
.L30:
	movq	64(%rsp), %rdi
	cmpq	%rbp, %rdi
	je	.L31
	call	_ZdlPv@PLT
.L31:
	movq	%r12, %rbp
.L32:
	movq	32(%rsp), %rdi
	cmpq	%r13, %rdi
	je	.L34
	call	_ZdlPv@PLT
.L34:
	movq	(%rsp), %rdi
	cmpq	%rbx, %rdi
	je	.L35
	call	_ZdlPv@PLT
.L35:
	movq	%rbp, %rdi
.LEHB11:
	call	_Unwind_Resume@PLT
.LEHE11:
	.cfi_endproc
.LFE1014:
	.section	.gcc_except_table
.LLSDAC1014:
	.byte	0xff
	.byte	0xff
	.byte	0x1
	.uleb128 .LLSDACSEC1014-.LLSDACSBC1014
.LLSDACSBC1014:
	.uleb128 .LEHB11-.LCOLDB3
	.uleb128 .LEHE11-.LEHB11
	.uleb128 0
	.uleb128 0
.LLSDACSEC1014:
	.section	.text.unlikely
	.text
	.size	_Z5greetNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEE, .-_Z5
	.section	.text.unlikely
	.size	_Z5greetNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEE.cold,
.LCOLDE3:
	.text
.LHOTE3:
	.hidden	DW.ref.__gxx_personality_v0
	.weak	DW.ref.__gxx_personality_v0
	.section	.data.rel.local.DW.ref.__gxx_personality_v0,"awG",@progbits,DW.
	.align 8
	.type	DW.ref.__gxx_personality_v0, @object
	.size	DW.ref.__gxx_personality_v0, 8
DW.ref.__gxx_personality_v0:
	.quad	__gxx_personality_v0
	.ident	"GCC: (Debian 10.2.0-19) 10.2.0"
	.section	.note.GNU-stack,"",@progbits
```

```
	.cfi_def_cfa_offset 24
	.cfi_offset 3, -24
	movabsq $4611686018427387903, %rbx
	movq	%rbx, %rax
	subq	$8, %rsp
	.cfi_def_cfa_offset 32
	subq	8(%rdi), %rax
	cmpq	$9, %rax
	jbe .L3
	movq	%rdi, %rbp
	movl	$10, %edx
	leaq	.LC1(%rip), %rsi
	call	_ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm
	movq	%rbx, %rax
	subq	8(%rbp), %rax
	cmpq	$9, %rax
	jbe .L3
	movl	$10, %edx
	leaq	.LC1(%rip), %rsi
	movq	%rbp, %rdi
	call	_ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm
	movq	%rbx, %rax
	subq	8(%rbp), %rax
	cmpq	$9, %rax
	jbe .L3
	movl	$10, %edx
	leaq	.LC1(%rip), %rsi
	movq	%rbp, %rdi
	call	_ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm
	movq	%rbx, %rax
	subq	8(%rbp), %rax
	cmpq	$9, %rax
	jbe .L3
	movl	$10, %edx
	leaq	.LC1(%rip), %rsi
	movq	%rbp, %rdi
	call	_ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm
	subq	8(%rbp), %rbx
	cmpq	$9, %rbx
	jbe .L3
	addq	$8, %rsp
	.cfi_remember_state
	.cfi_def_cfa_offset 24
	movq	%rbp, %rdi
	movl	$10, %edx
	popq	%rbx
	.cfi_def_cfa_offset 16
	leaq	.LC1(%rip), %rsi
	popq	%rbp
	.cfi_def_cfa_offset 8
	jmp _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@PLT
.L3:
	.cfi_restore_state
	leaq	.LC0(%rip), %rdi
	call	_ZSt20__throw_length_errorPKc@PLT
	.cfi_endproc
.LFE1014:
	.size	_Z5greetRNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE, .-_Z5
	.ident	"GCC: (Debian 10.2.0-19) 10.2.0"
	.section	.note.GNU-stack,"",@progbits
```

Returning values

```
void hello(std::string& val)
{
    val.append(”Hello C++!”);
}

void greet(std::string& s)
{
    hello(s);
    hello(s);
    hello(s);
    hello(s);
    hello(s);
}
```

```
std::string&& hello(std::string&& val)
{
    val.append(”Hello C++!”);
    return std::move(val);
}

std::string&& greet(std::string&& s)
{
    return hello(hello(hello(hello(hello(
        std::move(s))))));
}
```



```
    pushq   %rbx
    .cfi_def_cfa_offset 24
    .cfi_offset 3, -24
    movabsq $4611686018427387903, %rbx
    movq    %rbx, %rax
    subq    $8, %rsp
    .cfi_def_cfa_offset 32
    subq    8(%rdi), %rax
    cmpq    $9, %rax
    jbe .L3
    movq    %rdi, %rbp
    movl    $10, %edx
    leaq    .LC1(%rip), %rsi
    call    _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@
    movq    %rbx, %rax
    subq    8(%rbp), %rax
    cmpq    $9, %rax
    jbe .L3
    movl    $10, %edx
    leaq    .LC1(%rip), %rsi
    movq    %rbp, %rdi
    call    _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@
    movq    %rbx, %rax
    subq    8(%rbp), %rax
    cmpq    $9, %rax
    jbe .L3
    movl    $10, %edx
    leaq    .LC1(%rip), %rsi
    movq    %rbp, %rdi
    call    _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@
    movq    %rbx, %rax
    subq    8(%rbp), %rax
    cmpq    $9, %rax
    jbe .L3
    movl    $10, %edx
    leaq    .LC1(%rip), %rsi
    movq    %rbp, %rdi
    call    _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@
    subq    8(%rbp), %rbx
    cmpq    $9, %rbx
    jbe .L3
    addq    $8, %rsp
    .cfi_remember_state
    .cfi_def_cfa_offset 24
    movq    %rbp, %rdi
    movl    $10, %edx
    popq    %rbx
    .cfi_def_cfa_offset 16
    leaq    .LC1(%rip), %rsi
    popq    %rbp
    .cfi_def_cfa_offset 8
    jmp _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@PLT
.L3:
    .cfi_restore_state
    leaq    .LC0(%rip), %rdi
    call    _ZSt20__throw_length_errorPKc@PLT
    .cfi_endproc
.LFE1221:
    .size   _Z5greetRNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEE, .-_Z5g
    .ident  "GCC: (Debian 10.2.0-19) 10.2.0"
    .section    .note.GNU-stack,"",@progbits
```

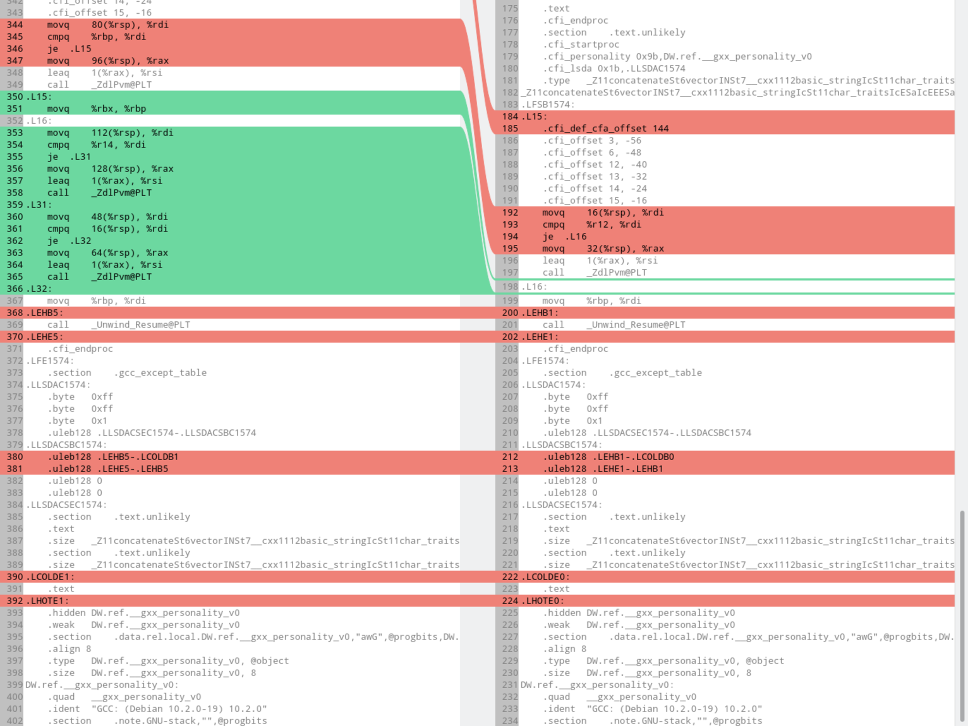
```
    pushq   %rbx
    .cfi_def_cfa_offset 24
    .cfi_offset 3, -24
    movabsq $4611686018427387903, %rbx
    movq    %rbx, %rax
    subq    $8, %rsp
    .cfi_def_cfa_offset 32
    subq    8(%rdi), %rax
    cmpq    $9, %rax
    jbe .L3
    movq    %rdi, %r12
    movl    $10, %edx
    leaq    .LC1(%rip), %rsi
    call    _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@
    movq    %rbx, %rax
    subq    8(%r12), %rax
    cmpq    $9, %rax
    jbe .L3
    movl    $10, %edx
    leaq    .LC1(%rip), %rsi
    movq    %r12, %rdi
    call    _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@
    movq    %rbx, %rax
    subq    8(%r12), %rax
    cmpq    $9, %rax
    jbe .L3
    movl    $10, %edx
    leaq    .LC1(%rip), %rsi
    movq    %r12, %rdi
    call    _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@
    movq    %rbx, %rax
    subq    8(%r12), %rax
    cmpq    $9, %rax
    jbe .L3
    movl    $10, %edx
    leaq    .LC1(%rip), %rsi
    movq    %r12, %rdi
    call    _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@
    subq    8(%r12), %rbx
    cmpq    $9, %rbx
    jbe .L3
    movq    %r12, %rdi
    movl    $10, %edx
    leaq    .LC1(%rip), %rsi
    call    _ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_appendEPKcm@
    addq    $8, %rsp
    .cfi_remember_state
    .cfi_def_cfa_offset 24
    movq    %r12, %rax
    popq    %rbx
    .cfi_def_cfa_offset 16
    popq    %r12
    .cfi_def_cfa_offset 8
    ret
.L3:
    .cfi_restore_state
    leaq    .LC0(%rip), %rdi
    call    _ZSt20__throw_length_errorPKc@PLT
    .cfi_endproc
.LFE1222:
    .size   _Z5greetONSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEE, .-_Z5g
```

Testing strings

```
template <typename It, typename T, typename F>
T accumulate(It first, It last, T init, F op)
{
    for (; first != last; ++first) {
        init = op(init, *first);
    }

    return init;
}

std::string concatenate(std::vector<std::string> xs)
{
    return accumulate(xs.cbegin(), xs.cend(), std::string{},
        [] (std::string acc, const std::string& x)
            -> std::string {
            return acc + x;
        });
}
```

Testing strings (C++20)

```
template <typename It, typename T, typename F>
T accumulate(It first, It last, T init, F op)
{
    for (; first != last; ++first) {
        init = op(std::move(init), *first);
    }

    return init;
}

std::string concatenate(std::vector<std::string> xs)
{
    return accumulate(xs.cbegin(), xs.cend(), std::string{},
        [] (std::string acc, const std::string& x)
            -> std::string {
            acc.append(x);
            return acc;
        });
}
```

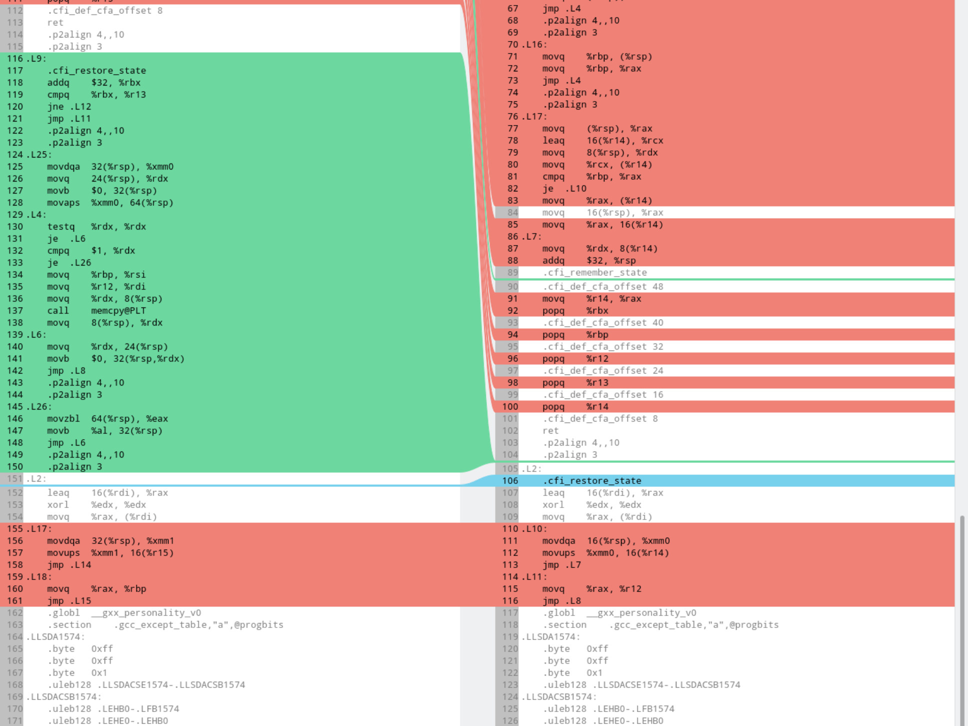


```
    .cfi_offset 14, -24
    .cfi_offset 15, -16
    movq    80(%rsp), %rdi
    cmpq    %rbp, %rdi
    je  .L15
    movq    96(%rsp), %rax
    leaq    1(%rax), %rsi
    call    _ZdlPvm@PLT
.L15:
    movq    %rbx, %rbp
.L16:
    movq    112(%rsp), %rdi
    cmpq    %r14, %rdi
    je  .L31
    movq    128(%rsp), %rax
    leaq    1(%rax), %rsi
    call    _ZdlPvm@PLT
.L31:
    movq    48(%rsp), %rdi
    cmpq    16(%rsp), %rdi
    je  .L32
    movq    64(%rsp), %rax
    leaq    1(%rax), %rsi
    call    _ZdlPvm@PLT
.L32:
    movq    %rbp, %rdi
.LEHB5:
    call    _Unwind_Resume@PLT
.LEHE5:
    .cfi_endproc
.LFE1574:
    .section    .gcc_except_table
.LLSDAC1574:
    .byte   0xff
    .byte   0xff
    .byte   0x1
    .uleb128 .LLSDACSEC1574-.LLSDACSBC1574
.LLSDACSBC1574:
    .uleb128 .LEHB5-.LCOLDB1
    .uleb128 .LEHE5-.LEHB5
    .uleb128 0
    .uleb128 0
.LLSDACSEC1574:
    .section    .text.unlikely
    .text
    .size   _Z11concatenateSt6vectorINSt7__cxx1112basic_stringIcSt11char_traits
    .section    .text.unlikely
    .size   _Z11concatenateSt6vectorINSt7__cxx1112basic_stringIcSt11char_traits
.LCOLDE1:
    .text
.LHOTE1:
    .hidden DW.ref.__gxx_personality_v0
    .weak   DW.ref.__gxx_personality_v0
    .section    .data.rel.local.DW.ref.__gxx_personality_v0,"awG",@progbits,DW.
    .align 8
    .type   DW.ref.__gxx_personality_v0, @object
    .size   DW.ref.__gxx_personality_v0, 8
DW.ref.__gxx_personality_v0:
    .quad   __gxx_personality_v0
    .ident  "GCC: (Debian 10.2.0-19) 10.2.0"
    .section    .note.GNU-stack,"",@progbits
```

```
    .text
    .cfi_endproc
    .section    .text.unlikely
    .cfi_startproc
    .cfi_personality 0x9b,DW.ref.__gxx_personality_v0
    .cfi_lsda 0x1b,.LLSDAC1574
    .type   _Z11concatenateSt6vectorINSt7__cxx1112basic_stringIcSt11char_traits
_Z11concatenateSt6vectorINSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEESa
.LFSB1574:
.L15:
    .cfi_def_cfa_offset 144
    .cfi_offset 3, -56
    .cfi_offset 6, -48
    .cfi_offset 12, -40
    .cfi_offset 13, -32
    .cfi_offset 14, -24
    .cfi_offset 15, -16
    movq    16(%rsp), %rdi
    cmpq    %r12, %rdi
    je  .L16
    movq    32(%rsp), %rax
    leaq    1(%rax), %rsi
    call    _ZdlPvm@PLT
.L16:
    movq    %rbp, %rdi
.LEHB1:
    call    _Unwind_Resume@PLT
.LEHE1:
    .cfi_endproc
.LFE1574:
    .section    .gcc_except_table
.LLSDAC1574:
    .byte   0xff
    .byte   0xff
    .byte   0x1
    .uleb128 .LLSDACSEC1574-.LLSDACSBC1574
.LLSDACSBC1574:
    .uleb128 .LEHB1-.LCOLDB0
    .uleb128 .LEHE1-.LEHB1
    .uleb128 0
    .uleb128 0
.LLSDACSEC1574:
    .section    .text.unlikely
    .text
    .size   _Z11concatenateSt6vectorINSt7__cxx1112basic_stringIcSt11char_traits
    .section    .text.unlikely
    .size   _Z11concatenateSt6vectorINSt7__cxx1112basic_stringIcSt11char_traits
.LCOLDE0:
    .text
.LHOTE0:
    .hidden DW.ref.__gxx_personality_v0
    .weak   DW.ref.__gxx_personality_v0
    .section    .data.rel.local.DW.ref.__gxx_personality_v0,"awG",@progbits,DW.
    .align 8
    .type   DW.ref.__gxx_personality_v0, @object
    .size   DW.ref.__gxx_personality_v0, 8
DW.ref.__gxx_personality_v0:
    .quad   __gxx_personality_v0
    .ident  "GCC: (Debian 10.2.0-19) 10.2.0"
    .section    .note.GNU-stack,"",@progbits
```

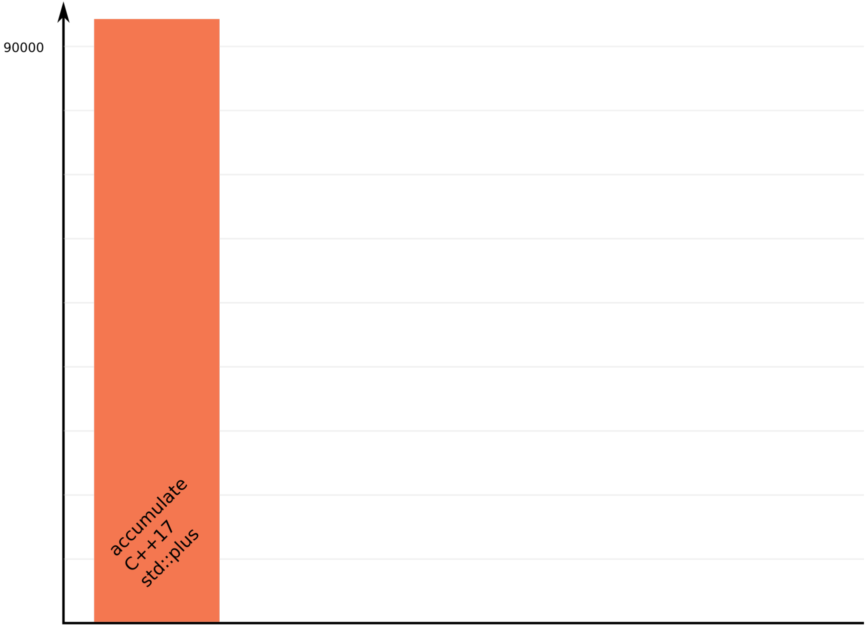
Testing strings (C++20)

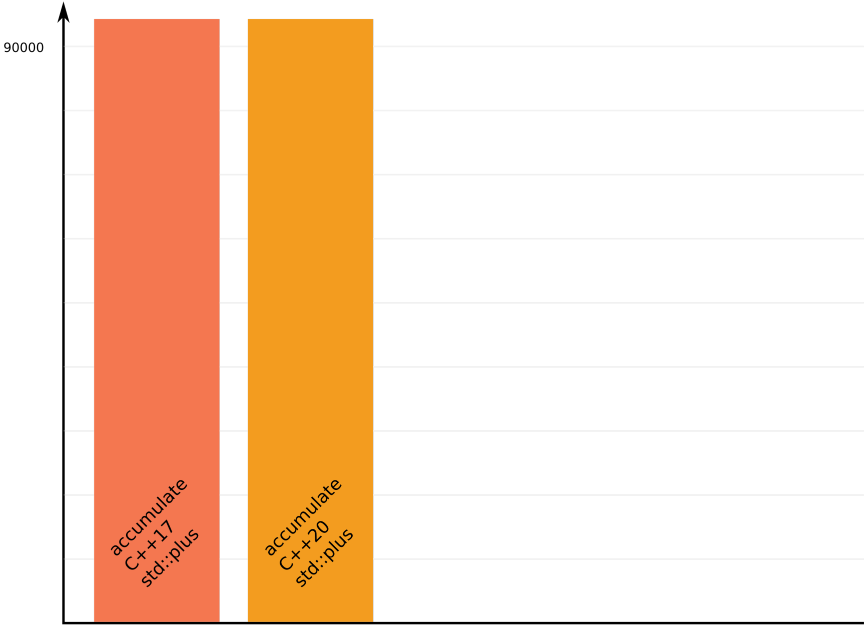
```
std::string concatenate(std::vector<std::string> xs)
{
    return accumulate(xs.cbegin(), xs.cend(), std::string{},
        [] (std::string&& acc, const std::string& x)
            -> std::string&& {
            acc.append(x);
            return std::move(acc);
        });
}
```

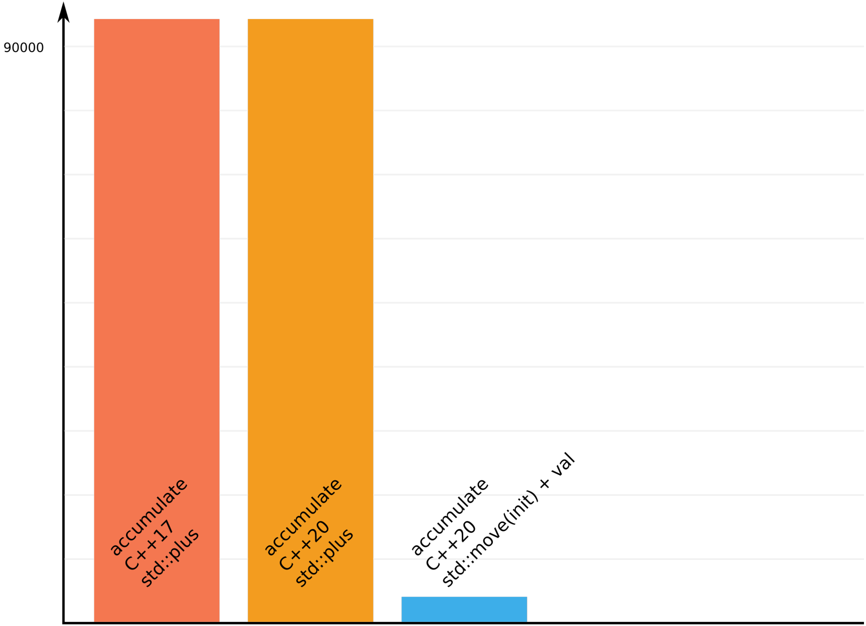


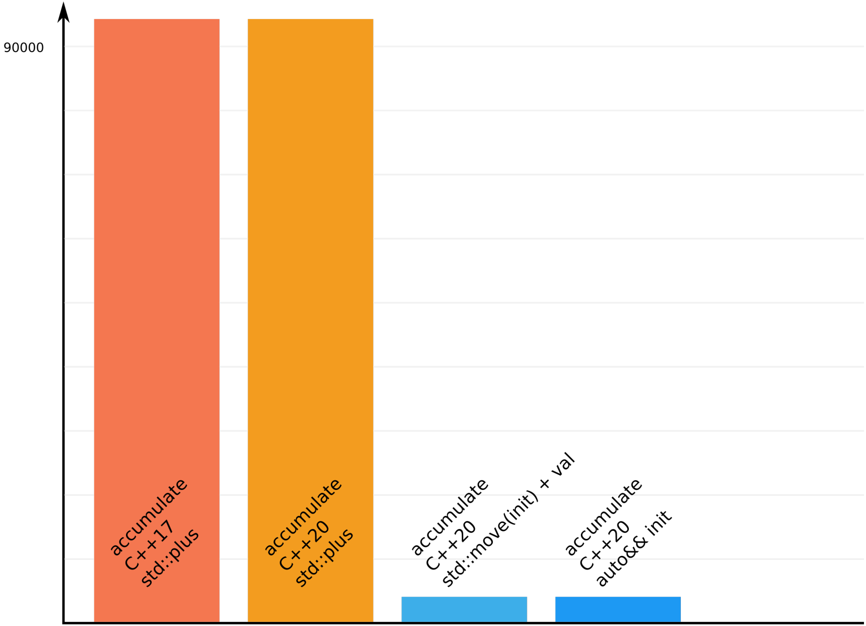
```
    .cfi_def_cfa_offset 8
    ret
    .p2align 4,,10
    .p2align 3
.L9:
    .cfi_restore_state
    addq    $32, %rbx
    cmpq    %rbx, %r13
    jne .L12
    jmp .L11
    .p2align 4,,10
    .p2align 3
.L25:
    movdqa  32(%rsp), %xmm0
    movq    24(%rsp), %rdx
    movb    $0, 32(%rsp)
    movaps  %xmm0, 64(%rsp)
.L4:
    testq   %rdx, %rdx
    je  .L6
    cmpq    $1, %rdx
    je  .L26
    movq    %rbp, %rsi
    movq    %r12, %rdi
    movq    %rdx, 8(%rsp)
    call    memcpy@PLT
    movq    8(%rsp), %rdx
.L6:
    movq    %rdx, 24(%rsp)
    movb    $0, 32(%rsp,%rdx)
    jmp .L8
    .p2align 4,,10
    .p2align 3
.L26:
    movzbl  64(%rsp), %eax
    movb    %al, 32(%rsp)
    jmp .L6
    .p2align 4,,10
    .p2align 3
.L2:
    leaq    16(%rdi), %rax
    xorl    %edx, %edx
    movq    %rax, (%rdi)
.L17:
    movdqa  32(%rsp), %xmm1
    movups  %xmm1, 16(%r15)
    jmp .L14
.L18:
    movq    %rax, %rbp
    jmp .L15
    .globl  __gxx_personality_v0
    .section    .gcc_except_table,"a",@progbits
.LLSDA1574:
    .byte   0xff
    .byte   0xff
    .byte   0x1
    .uleb128 .LLSDACSE1574-.LLSDACSB1574
.LLSDACSB1574:
    .uleb128 .LEHB0-.LFB1574
    .uleb128 .LEHE0-.LEHB0
```

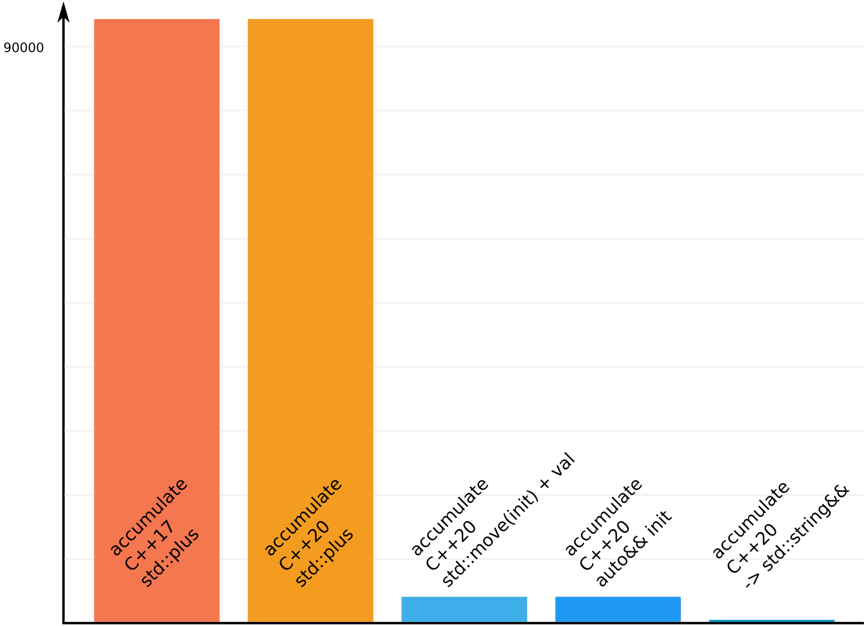
```
    jmp .L4
    .p2align 4,,10
    .p2align 3
.L16:
    movq    %rbp, (%rsp)
    movq    %rbp, %rax
    jmp .L4
    .p2align 4,,10
    .p2align 3
.L17:
    movq    (%rsp), %rax
    leaq    16(%r14), %rcx
    movq    8(%rsp), %rdx
    movq    %rcx, (%r14)
    cmpq    %rbp, %rax
    je  .L10
    movq    %rax, (%r14)
    movq    16(%rsp), %rax
    movq    %rax, 16(%r14)
.L7:
    movq    %rdx, 8(%r14)
    addq    $32, %rsp
    .cfi_remember_state
    .cfi_def_cfa_offset 48
    movq    %r14, %rax
    popq    %rbx
    .cfi_def_cfa_offset 40
    popq    %rbp
    .cfi_def_cfa_offset 32
    popq    %r12
    .cfi_def_cfa_offset 24
    popq    %r13
    .cfi_def_cfa_offset 16
    popq    %r14
    .cfi_def_cfa_offset 8
    ret
    .p2align 4,,10
    .p2align 3
.L2:
    .cfi_restore_state
    leaq    16(%rdi), %rax
    xorl    %edx, %edx
    movq    %rax, (%rdi)
.L10:
    movdqa  16(%rsp), %xmm0
    movups  %xmm0, 16(%r14)
    jmp .L7
.L11:
    movq    %rax, %r12
    jmp .L8
    .globl  __gxx_personality_v0
    .section    .gcc_except_table,"a",@progbits
.LLSDA1574:
    .byte   0xff
    .byte   0xff
    .byte   0x1
    .uleb128 .LLSDACSE1574-.LLSDACSB1574
.LLSDACSB1574:
    .uleb128 .LEHB0-.LFB1574
    .uleb128 .LEHE0-.LEHB0
```

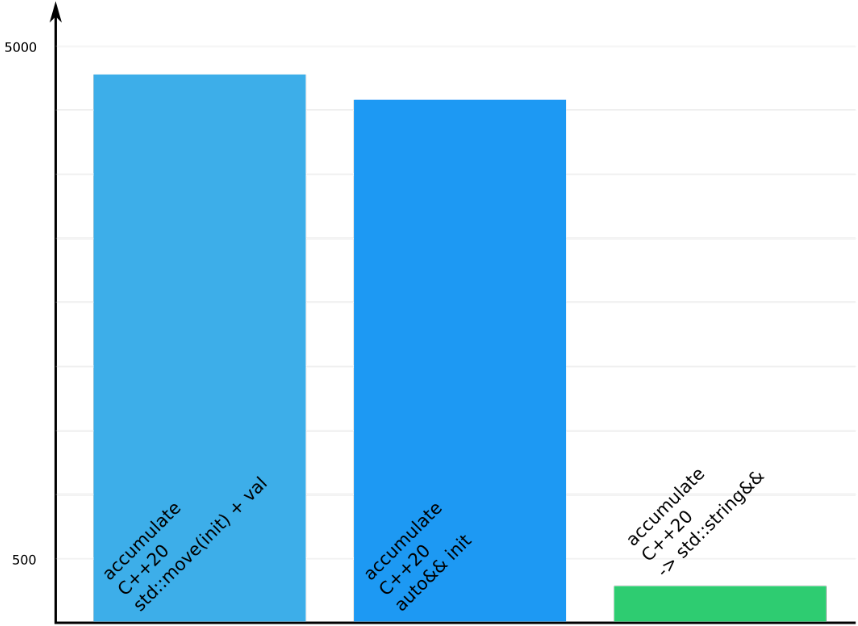

90000
accumulate
C++17
std::plus


90000
accumulate C++17 std::plus
accumulate C++20 std::plus


90000
accumulate
C++17
std::plus
accumulate
C++20
std::plus
accumulate
C++20
std::move(init) + val


90000
accumulate
C++17
std::plus
accumulate
C++20
std::plus
accumulate
C++20
std::move(init) + val
accumulate
C++20
auto&& init


90000
accumulate C++17 std::plus
accumulate C++20 std::plus
accumulate C++20 std::move(init) + val
accumulate C++20 auto&& init
accumulate C++20 -> std::string&&


5000
500
accumulate
C++20
std::move(init) + val
accumulate
C++20
auto&& init
accumulate
C++20
-> std::string&&

SAFETY

Safety

- Arguments by &&-reference
- Return by &&-reference
- Store results by &&-reference

Safety

All temporary objects are destroyed as the last step in evaluating the full-expression that (lexically) contains the point where they were created.

If multiple temporary objects were created, they are destroyed in the order opposite to the order of creation.

Safety

```
type&& f(type&& t)
{
    ...
}
```

Safe if `t` (or a global object) is returned in all paths.

Safety

```
type&& f()
{
    ...
}
```

Unsafe unless returning something global (external to `f`).

Safety

```
converted_type&& f(type&& t)
{
    ...
}
```

Unsafe, return by value.

Note: `explicit` is your friend.

Safety

- Consider returning &&
- But be cautious of dangling references
- Store result by-value

Safety

```
for (auto x: foo().value()) {
}
```

Safety

```
for (auto f = foo(); auto x: f.value()) {
}
```

Safety

- Use after move
  (`clang-tidy:bugprone-use-after-move`)
- Unused variable error
  (`-Werror=unused-variable`)
- Error handling
  (`optional<T>, expected<T,E>`)

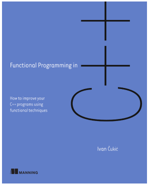
Answers? Questions! Questions? Answers!

Reaching me

Web: https://**cukic.co**
Mail: **ivan@cukic.co**
Twitter: **@ivan_cukic**

@KDAB

Web: https://**kdab.com**
Mail: **ivan.cukic@kdab.com**



Functional Programming in C++

How to improve your C++ programs using functional techniques

Ivan Čukić

MANNING

cukic.co/to/fp-in-cpp
Functional Programming in C++