Iteration using Indexing

The C++ source code was:

```
int main() {
char* arr = { "iteration"}; int i = 0; while (arr[i]) ++i;
}
```

Where the corresponding assembly code given by `objdump -d` was:

```
00000000004004d6 <main>:
4004d6:  55 push %rbp
4004d7:  48 89 e5 mov %rsp,%rbp
4004da:  48 c7 45 f8 94 05 40 movq $0x400594,-0x8(%rbp)
4004e1:  00
4004e2:  c7 45 f4 00 00 00 00 movl $0x0,-0xc(%rbp)
4004e9:  8b 45 f4 mov -0xc(%rbp),%eax
4004ec:  48 63 d0 movslq %eax,%rdx
4004ef:  48 8b 45 f8 mov -0x8(%rbp),%rax
4004f3:  48 01 d0 add %rdx,%rax
4004f6:  0f b6 00 movzbl (%rax),%eax
4004f9:  84 c0 test %al,%al
4004fb:  74 06 je 400503 <main+0x2d>
4004fd:  83 45 f4 01 addl $0x1,-0xc(%rbp)
400501:  eb e6 jmp 4004e9 <main+0x13>
400503:  b8 00 00 00 00 mov $0x0,%eax
400508:  5d pop %rbp
400509:  c3 retq
40050a:  66 0f 1f 44 00 00 nopw 0x0(%rax,%rax,1)
```

Using `-O2` optimization

```
00000000004003e0 <main>:
4003e0:  31 c0 xor %eax,%eax
4003e2:  c3 retq
4003e3:  66 2e 0f 1f 84 00 00 nopw %cs:0x0(%rax,%rax,1)
4003ea:  00 00 00
4003ed:  0f 1f 00 nopl (%rax)
```

Iteration using Pointers

The C++ source code was:

```
int main() {
char* arr = { "iteration"}; char* p = arr; while (*p) ++p;
}
```

Where the corresponding assembly code given by `objdump -d` was:

```
00000000004004d6 <main>:
4004d6:  55 push %rbp
4004d7:  48 89 e5 mov %rsp,%rbp
4004da:  48 c7 45 f8 94 05 40 movq $0x400594,-0x8(%rbp)
4004e1:  00
4004e2:  48 8b 45 f8 mov -0x8(%rbp),%rax
4004e6:  48 89 45 f0 mov %rax,-0x10(%rbp)
4004ea:  48 8b 45 f0 mov -0x10(%rbp),%rax
4004ee:  0f b6 00 movzbl (%rax),%eax
4004f1:  84 c0 test %al,%al
4004f3:  74 07 je 4004fc <main+0x26>
4004f5:  48 83 45 f0 01 addq $0x1,-0x10(%rbp)
4004fa:  eb ee jmp 4004ea <main+0x14>
4004fc:  b8 00 00 00 00 mov $0x0,%eax
400501:  5d pop %rbp
400502:  c3 retq
400503:  66 2e 0f 1f 84 00 00 nopw %cs:0x0(%rax,%rax,1)
40050a:  00 00 00
40050d:  0f 1f 00 nopl (%rax)
```

Using `-O2` optimization

```
00000000004003e0 <main>:
4003e0:  b8 84 05 40 00 mov $0x400584,%eax
4003e5:  0f 1f 00 nopl (%rax)
4003e8:  48 83 c0 01 add $0x1,%rax
4003ec:  80 38 00 cmpb $0x0,(%rax)
4003ef:  75 f7 jne 4003e8 <main+0x8>
4003f1:  31 c0 xor %eax,%eax
4003f3:  c3 retq
4003f4:  66 2e 0f 1f 84 00 00 nopw %cs:0x0(%rax,%rax,1)
4003fb:  00 00 00
4003fe:  66 90 xchg %ax,%ax
```

Conclusions

According to

```
dt@VPCF11M1E  /2019/c++11/exercises/08_pointers_arrays_and_references/13
$ diff -y --suppress-common-lines solution/pointered.asm solution/indexed.asm
```

and comparing `*_optimized.asm`'s using `-O2`:

1. The compiler will not generate equivalent code for iterating with indices vs. iterating with pointers.
2. Using the optimizer will radically reduce the size of the generated code, where *iterating with indices is still cheaper* than iterating with pointers.