ETH zürich

Informatik

Übungsstunde Woche 11

Prüfungsaufgaben

Gegeben `int x`, welcher der folgenden Ausdrücke ist KEIN L-Wert? / Given `int x`, which of the following expressions is NOT an L-value?

- ○ `++x` 
- ○ `&x` 
- ○ `x += 1` 
- ○ `x` 

Prüfungsaufgaben

Gegeben `int x`, welcher der folgenden Ausdrücke ist KEIN L-Wert? / Given `int x`, which of the following expressions is NOT an L-value?

✖ `++x`	S	
✔ `&x`	S	✔
✖ `x += 1`	S	
✖ `x`	S	

Prüfungsaufgaben

Welche const-Deklaration erlaubt p++? / Which const-declaration allows for p++?

- ○ `int const p;`
- ○ `int const* const p;`
- ○ `int* const p;`
- ○ `int const* p;`

Prüfungsaufgaben

Welche const-Deklaration erlaubt p++? / Which const-declaration allows for p++?

✗ `int const p;`	S	
✗ `int const* const p;`	S	
✗ `int* const p;`	S	
✓ `int const* p;`	S	✓

Prüfungsaufgaben

```
1  int sum_swap(int& x, int y) {
2    int temp = x;
3    x = y;
4    y = temp;
5    return x + y;
6  }
7
8  int a = 3;
9  int b = 5;
10 int c = sum_swap(a, b);
```

Welche Werte haben a, b und c am Ende des Codeschnippsels? / What are the final values of a, b and c?

a =

b =

c =

Prüfungsaufgaben

```
1  int sum_swap(int& x, int y) {
2    int temp = x;
3    x = y;
4    y = temp;
5    return x + y;
6  }
7
8  int a = 3;
9  int b = 5;
10 int c = sum_swap(a, b);
```

Welche Werte haben a, b und c am Ende des Codeschnippsels? / What are the final values of a, b and c?

a = 5 ✔

b = 5 ✔

c = 8 ✔

Prüfungsaufgaben

```
1 int* arr = new int[4] {2, 1, 3, 0};
2 int* x = arr + 1;
3 int* y = (&arr[2]) - 2;
4
5 std::cout << *(x + *y);
```

Welche Aussage beschreibt die Ausgabe am besten? / Which statement describes the output best?

- ○ 0
- ○ 1
- ○ Fehler (Compiler- oder Laufzeit-) / Error (compiler or runtime)
- ○ 2
- ○ 3

Prüfungsaufgaben

```
1 int* arr = new int[4] {2, 1, 3, 0};
2 int* x = arr + 1;
3 int* y = (&arr[2]) - 2;
4
5 std::cout << *(x + *y);
```

Welche Aussage beschreibt die Ausgabe am besten? / Which statement describes the output best?

- ✔ 0 ✔
- ✖ 1
- ✖ Fehler (Compiler- oder Laufzeit-) / Error (compiler or runtime)
- ✖ 2
- ✖ 3

Pointer Program

```
int* a = new int[5]{0, 8, 7, 2, -1};
int* ptr = a;                          // pointer assignment
++ptr;                                 // shift to the right
int my_int = *ptr;                     // read target
ptr += 2;                              // shift by 2 elements
*ptr = 18;                             // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```

Pointer Program

Find and fix at least 3 problems in the following program.

```
#include <iostream>
int main () {
    int* a = new int[7]{0, 6, 5, 3, 2, 4, 1};
    int* b = new int[7];
    int* c = b;
    // copy a into b using pointers
    for (int* p = a; p <= a+7; ++p) {
        *c++ = *p;
    }
    // cross-check with random access
    for (int i = 0; i <= 7; ++i) {
        if (a[i] != c[i]) {
            std::cout << "Oops, copy error...\n";
        }
    }
    return 0;
}
```

(From: Script Exercise 117)

Pointer Program

```
#include <iostream>
int main () {
    int* a = new int[7]{0, 6, 5, 3, 2,
    int* b = new int[7];
    int* c = b;
    // copy a into b using pointers
    for (int* p = a; p <= a+7; ++p) {
        *c++ = *p;
    }
    // cross-check with random access
    for (int i = 0; i <= 7; ++i) {
        if (a[i] != c[i]) {
            std::cout << "Oops, copy error...\n";
        }
    }
    return 0;
}
```

p = a+7 is dereferenced

Solution:
Use < instead of <=

(From: Script Exercise 117)

Pointer Program

```
#include <iostream>
int main () {
    int* a = new int[7]{0, 6, 5, 3, 2, 4
    int* b = new int[7];
    int* c = b;
    // copy a into b using pointers
    for (int* p = a; p <= a+7; ++p) {
        *c++ = *p;
    }
    // cross-check with random access
    for (int i = 0; i <= 7; ++i) {
        if (a[i] != c[i]) {
            std::cout << "Oops, copy error
        }
    }
    return 0;
}
```

p = a+7 is dereferenced

Solution:
Use < instead of <=

Same problem as above

(From: Script Exercise 117)

Pointer Program

```
#include <iostream>
int main () {
    int* a = new int[7]{0, 6, 5, 3, 2, 4, 1};
    int* b = new int[7];
    int* c = b;
    // copy a into b using pointers
    for (int* p = a; p <= a+7; ++p) {
        *c++ = *p;

    // cross-check with random access
    for (int i = 0; i <= 7; ++i) {
        if (a[i] != c[i]) {
            std::cout << "Oops, copy err
        }
    }
    return 0;
}
```

`c` doesn't point to `b[0]` anymore.

Solution:
Use `b` instead of `c`

`p = a+7` is dereferenced

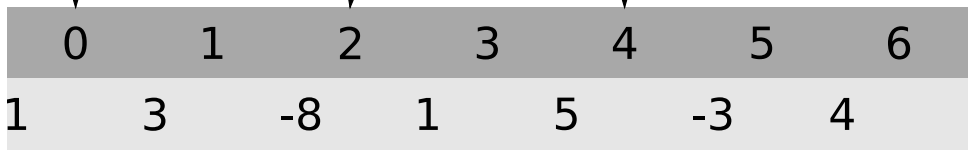
Solution:
Use `<` instead of `<=`

Same problem as above

Exercise – Applying Pointers

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint
//      valid ranges
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

Variable	Value
b	●
e	●
o	●



(From: Script Exercise 113)

Exercise – Applying Pointers

Now determine a POST-condition for the function.

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint
//      valid ranges
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

(From: Script Exercise 113)

Exercise – Applying Pointers

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint
//      valid ranges
// POST: The range [b, e) is copied in reverse
//       order into the range [o, o+(e-b))
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

(From: Script Exercise 113)

Exercise – Valid Inputs

- Which of these inputs are valid?

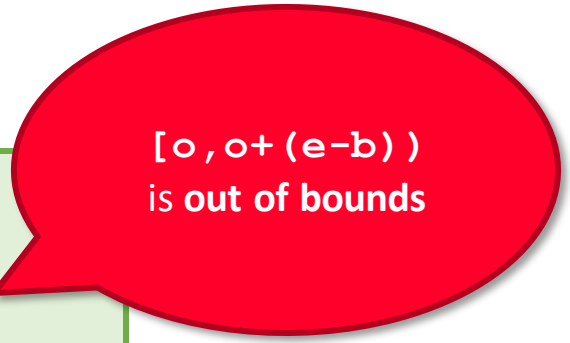
```
int* a = new int[5];
// Initialise a.
  a)  f(a, a+5, a+5);
  b)  f(a, a+2, a+3);
  c)  f(a, a+3, a+2);
```

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint
//      valid ranges
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

Exercise – Valid Inputs

- Which of these inputs are valid?

```
int* a = new int[5];
// Initialise a.
  a)  f(a, a+5, a+5);
  b)  f(a, a+2, a+3);
  c)  f(a, a+3, a+2);
```




```
// PRE: [b, e) and [o, o+(e-b)) are disjoint
//      valid ranges
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

(From: Script Exercise 113)

Exercise – Valid Inputs

- Which of these inputs are valid?

```
int* a = new int[5];
// Initialise a.
  a)  f(a, a+5, a+5);
  b)  f(a, a+2, a+3);
  c)  f(a, a+3, a+2);
```

[o, o+(e-b)) is **out of bounds**

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint
//      valid ranges
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

(From: Script Exercise 113)

Exercise – Valid Inputs

- Which of these inputs are valid?

```
int* a = new int[5];
// Initialise a.
  a)  f(a, a+5, a+5);
  b)  f(a, a+2, a+3);
  c)  f(a, a+3, a+2);
```

[o,o+(e-b))
is **out of bounds**

Ranges **not disjoint**

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint
//      valid ranges
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

(From: Script Exercise 113)

Exercise – `const` Correctness

- Make the function `const`-correct.

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint
//      valid ranges
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

(From: Script Exercise 113)

Exercise – `const` Correctness

- Make the function `const`-correct.

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint
//      valid ranges
void f (const int* const b, const int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

(From: Script Exercise 113)