10 Control Flow - Repetition

Monday, October 19, 2020 9:23 AM

* Read § 6.5 and 6.6 from Scott
* Assignment #3 Presentations on Oct 26 (Monday)

C, C++, Java

pre-test loop

```
while ( condition ) {
    statements
}
```

Logically controlled

```
do {
    statements
} while ( condition );
```

post test loop

```
for ( init; condition; update ) {
    statements
}
```

combination loop or enumeration loop

Python

```
while condition:
    statements
```

```
for variable in sequence:
    statements
```

Anatomy of Loop

1. loop index / loop control variable
2. loop condition
3. loop update
4. loop body

index, condition, update

```
for (int i = 0; i < n; i++) {
    statements
}
```

```
int i = 0;
while ( i < n) {
    statements
    i = i + 1;
}
```

- break
- continue

Other Loop Designs

mid-test loop

```
while ( 1 ) {
    ...
    if ( condition )
        break;
```

Infinite Loops

```
while True:
    ...
    if condition:
        break
```

```
for ( ; ; ; ) {
    ...
    if ( condition )
        break;
```

loop

```
…
if ( condition )
    break;
…
```

C/C++/Java

```
…
if condition:
    break
…
```

Python

```
…
if ( condition )
    break;
…
```

C/C++/Java

10 Control Flow - Loop Design Issues

Monday, October 19, 2020 8:25 AM

Other Languages

```
for i := 0 to n-1 do
begin
      statements
end
```

count up

```
for i := n-1 downto 0 do
begin
      statements
end
```

count down

Pascal

```
FOR i := 0 TO n-1 DO
    statements
END
```

```
FOR i:= n-1 TO 0 BY -1 DO
     statements
END
```

Modula-2

Loop Design Issues

- Loop control variable (not required in a while-loop. Also optional in for-loop.)
- Scope of the LCV

```
int i;
for (i=0; i<n; i++){
    |
}
// i is visible here.
```

```
for (int i=0; i<n; i++){
    |
}
// i is not available here...
```

- Can the LCV be modified in the loop?

```
for (int i=0; i<10; i++){
    "println"(i)
    i = i+1;
}
```

0 1 2 3 4 5 6 7 8 9

0 2 4 6 8

Python

```
for i in range(10):
    print(i)
    i = i+1
```

0 2 3 4 5 6 7 8 9

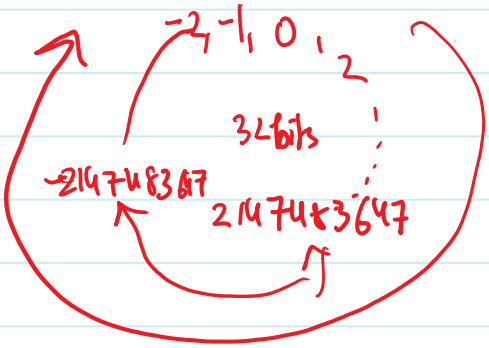
range(10) → [0,1,2,3,4,5,6,7,8,9]

10 Control Flow - Loop Design Issues, *contd*.

Monday, October 19, 2020 8:25 AM

- What happens at edge cases? $2^{31}-1 =$

```
for (int i=__ ; i <=2147483647; i++){
    |
}
```



$$2147483646 + 1 = 2147483647$$
$$2147483647 + 1 = 2147483648 \equiv -2147483648$$

C/C++: INT_MIN, INT_MAX
Java: Integer.MAX_VALUE
Integer.MIN_VALUE

- Does the PL allow iteration over user-defined types??

```
BinaryTree<T> t;
...
for (T item : t){
    | // do something with item
}
```

Iterator

```
public class BinaryTree<T>
        implements Iterator{
    }
```

```
int[] a = ...
for (int item : a){
    | // can use the value of a
    | // but not modify it.
}
```

```
ArrayList<Place> places = .....
for (Place p : places){
    |
}
```

10 Control Flow - Iterators

Monday, October 19, 2020 8:25 AM

names = ["Bryn Mawr", "Haverford", "Swarthmore", ...]

for name in names:

word = "Bryn Mawr"

for w in word:
    print(w)

for <variable> in <sequence>:

Python Iterators are co-routines.

Java: User-defined objects/classes have to implement their own iterators:

1. Implement Iterable interface

```
public Iterator iterator();
```

2. Interator interface

```
public boolean hasNext();
public E next();
public void remove();
```

ArrayList<Place> places ....

```
Iterator<Place> iter = places.iterator();
while (iter.hasNext()) {
    Place p = iter.next();
}
```

```
for (Place p: places) {
}
```

10 Control Flow - Recursion

Monday, October 19, 2020 8:25 AM

Iterative code

Recursive

$$Sum = \sum_{i=1}^{n} i$$

```
int sum(int low, int high) {
    int result = 0;
    for (int i=low; i <= high; i++)
        result = result + i;
    returnresult;
} // sum()
```

```
int sum(int low, int high) {
    if (low == high)
        return low;
    else
        return low + sum(low+1, high);
} // sum()
```

```
int gcd (int a, int b) {
    while (a != b) {
        if (a > b)
            a = a - b;
        else
            b = b - 1;
    }
    return a;
} // gcd()
```

```
int gcd(int a, int b) {
    if (a == b)
        return a;
    else if (a > b)
        return gcd(a-b, b);
    else
        return gcd(a, b-a);
} // gcd()
```

0! = 1

n! = n * (n-1)!

```
int factorial(int n) {
    int result = 1;
    for (int i=1; i <= n; i++)
        result *= i;
    return result;
} // factorial()
```

```
int factorial(int n) {
    if (n == 0)
        return 1;
    else
        return n * factorial(n-1);
} // facorial()
```

Tail Recursion Tail Call optimization (TCO)

10 Control Flow - Iteration *vs* Recursion

Monday, October 19, 2020 8:25 AM

factorial(4)