CMSC 124

Design and Implementation of Programming Languages

CNM Peralta

CONTROL STRUCTURES

Control Statements

Allow the **selection** of one of possibly many different control flow paths or the **repeated execution** of a [sequence of] statements.

Research in the **mid-1960s** to **mid-1970s** concluded that only **two kinds of control statements** are needed to **express any algorithm** that can be **outlined in a flowchart**.

1.

A control statement that can **choose between two control flow paths**.

2.

A control statement for **logically-controlled iterations**.

Control Structure

A **control statement** and the **collection of statements** whose **execution** it **controls**.

Example.

```
if(x % 2 == 0) {
  printf(“x is even.\n”);
} else {
  printf(“x is odd.\n”);
}
```

What is the **control statement**?

What is the **control structure**?

SELECTION STATEMENTS

Selection Statement

Chooses one of **two or more execution paths** in a program.

1.

Two-way selection statements

General Form

```
if control_expression
   then clause
   else clause
```

Design Issues

1.1.

What **form** will the **control expression** take?

1.2.

How are the `then-` and `else-` **clause specified**?

1.3.

How should the **meaning** of **nested selectors** be specified?

Usually, only **Boolean expressions** can be used as **control expressions**.

```
  x == 0 && y == 0

             x == 0 || y == 0
```

Exception

Languages such as **C**, **Python**, and **C++** allow the use **arithmetic expressions** as **control expressions**; the convention is: non-zero values are true, zero values are false.

```
int i = 10;
while(i-=2) {
  printf(“%d\n”, i);
}
```

`then`- and `else`-**clauses** can be **single** or **compound statements**.

```
if(x %2 == 0)
  printf("x is even.\n");
else
  printf("x is odd.\n");
```

Single statement

Compound statement

```
if(x %2 == 0) {
  printf("x is even.\n");
}
else {
  printf("x is odd.\n");
}
```

Some PLs use **statement sequences** instead of compound statements, and **end** their if-else statements with a **keyword/phrase**.

Example: Ada

```
if condition then
    statement;
else
    other statement;
end if;
```

Consider…

Java allows this:

```
if(sum == 0)
  if(count == 0)
    result = 0;
else
  result = 1;
```

This looks an awful lot like the **dangling-else problem** from out discussion on **ambiguity**.

General Static Semantic Rule for Two-waySelection Statements

The `else`-**clause** is always **paired** with the **nearest**, **previous**, **unpaired** `then`-**clause**.

To **force pair** the `else`-clause **to another `then`-clause**, use **compound statements** instead.

Thus,

```
if(sum == 0) {
 if(count == 0)
  result = 0;
} else
 result = 1;
```

QUIZ

2.

Multiple selection statements

Multiple-Selection Statement

Allow the **selection** of **one of any number** of **statements** or **statement groups**.

They are **generalized** selection statements.

Although it is possible to **build multiple-selection statements from two-way selection statements**, it will be **difficult** and possibly **unreliable**.

Example.

```
scanf(“%d”, &x);
switch(x) {
   case 1: //statements
   case 2: //statements
   ...
   default:
      //statements
}
```

vs

```
scanf(“%d”, &x);
if(x == 1) {
   //statements
}
if(x == 2) {
   //statements
}
...
//statements
```

Design Issues

1.

What is the **form** and **type** of the **expression** that **controls selection**?

Example: C

```
switch(x) {
   case 1:
   case 2:
   ...
}
```

The 'form' of expression that controls selection is an **equality test** between a **variable** (x) and the **various case values** (1, 2, etc.).

2.

How are the **selectable statements specified**?

Example: C

```
switch(x) {
   case 1:
   case 2:
   ...
}
```

The case keyword specifies selectable statements.

3.

Is **execution flow** through the structure **restricted** to include just a **single selectable statement**?

Example: C

```
switch(x) {
   case 1:
   case 2:
   ...
}
```

The **absence** of `break` **statements** allow the **execution** of **more than one case**.

4.

How are the **case values specified**?

Example: C

```
switch(x) {
   case 1:
   case 2:
   ...
}
```

In C, only **integers** and **characters** can be used in a `case` statement.

5.

How should **unrepresented selector expression values** be **handled**, if at all?

Example: C

```
switch(x) {
  case 1:
  case 2:
  ...
  default:
  ...
}
```

In C, the `default` case represents all values that are not used in the specific cases.

Implementing Selection Statements

Example: LOLCode

```
HAI
I HAS A VAR1
VISIBLE “Enter a number: ”
GIMMEH VAR1
BOTH SAEM MOD OF VAR1 AN 5 AND 0
O RLY?
YA RLY
  VISIBLE “Divisible by 5”
NO WAI
  VISIBLE “Not divisible by 5”
OIC
KTHXBYE
```

First, generate the list of lexemes and tags.

Index	Lexeme	Tag
0	`HAI`	Code delimiter
1	`I HAS A`	Variable Declaration Keyword
2	`VAR1`	Variable Identifier
3	`VISIBLE`	Output Statement Keyword
4	`“ENTER A NUMBER: ”`	Yarn Literal
5	`GIMMEH`	Input Statement Keyword
6	`VAR1`	Variable Identifier
7	`BOTH SAEM`	Equality Comparison Operator
8	`MOD OF`	Modulo Operator
9	`VAR1`	Variable Identifier
10	`AN`	Operand Separator
11	`5`	Numbr Literal
12	`AN`	Operand Separator
13	`0`	Numbr Literal
14	`O RLY?`	Selection Statement Keyword

Index	Lexeme	Tag
15	`YA RLY`	Selection Statement If-Clause Keyword
16	`VISIBLE`	Output Statement Keyword
17	`“Divisible by 5”`	Yarn Literal
18	`NO WAI`	Selection Statement Else-Clause Keyword
19	`VISIBLE`	Output Statement Keyword
20	`“Not divisible by 5”`	Yarn Literal
21	`OIC`	Selection Statement End Keyword
22	`KTHXBYE`	Code Delimiter

Once the `O RLY?` is encountered, you can **immediately look for the index** of the `if/then`**-clause**, `else`**-clause**, and the `OIC` keyword.

Clause	**Index**
`YA RLY`	15
`NO WAI`	18
`OIC`	21

Recall the specs: **bare expressions** have their **result** saved to the **implicit** `IT` **variable**.

```
HAI
I HAS A VAR1
VISIBLE “Enter a number: ”
GIMMEH VAR1
BOTH SAEM MOD OF VAR1 AN 5 AND 0
O RLY?
YA RLY
  VISIBLE “Divisible by 5”
NO WAI
  VISIBLE “Not divisible by 5”
OIC
KTHXBYE
```

Result stored in IT

```
HAI
I HAS A VAR1
VISIBLE “Enter a number: ”
GIMMEH VAR1
BOTH SAEM MOD OF VAR1 AN 5 AND 0
O RLY?
YA RLY
  VISIBLE “Divisible by 5”
NO WAI
  VISIBLE “Not divisible by 5”
OIC
KTHXBYE
```

Check value of `IT`; if true, jump to index of `YA RLY`, otherwise, jump to index of `NO WAI`.

```
HAI
I HAS A VAR1
VISIBLE “Enter a number: ”
GIMMEH VAR1
BOTH SAEM MOD OF VAR1 AN 5 AND 0
O RLY?
YA RLY
  VISIBLE “Divisible by 5”
NO WAI
  VISIBLE “Not divisible by 5”
OIC
KTHXBYE
```

If the `YA RLY` clause is executed, jump to the index of `OIC` afterward.

In the case of **nested** `if-else` **statements**, remember: a **stack** is a CMSC 124 student's best friend.

A similar strategy can be used for `WTF?` **statements**: find all the **indices** of `OMG` **cases** and the `OMGWTF` **case**, and the `OIC` **keyword**. Go through all the cases by using the indices; when a `GTFO` **statement** is encountered, jump to the `OIC` index.

Hint

C# also supports `structs`, if you do not wish to make make classes all the time.